

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

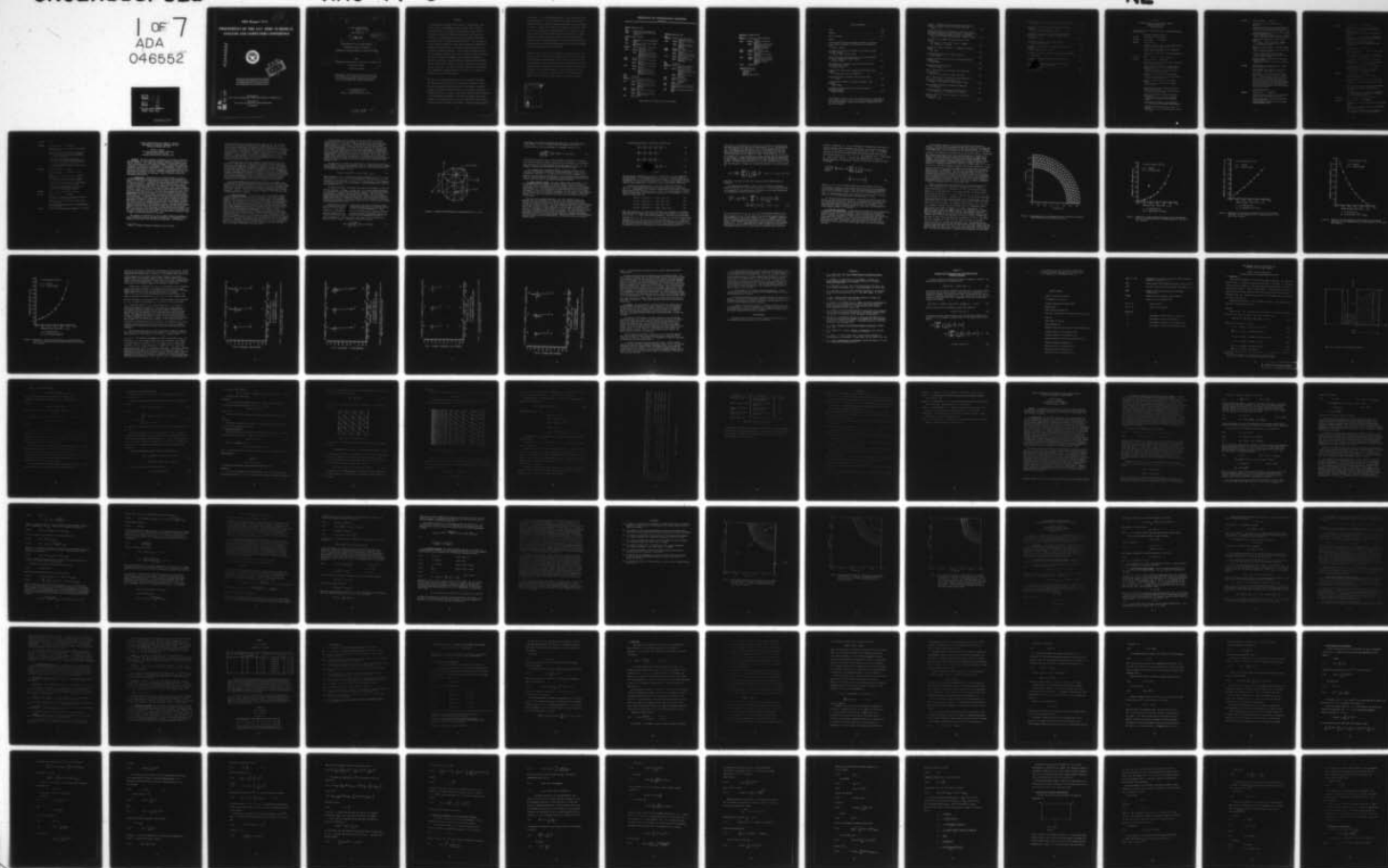
F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

1 of 7
ADA
046552



11024
7912

(12) 2

ARO Report 77-3

PROCEEDINGS OF THE 1977 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE

AD A 0 46552



Approved for public release; distribution unlimited.
The findings in this report are not to be construed
as an official Department of the Army position, un-
less so designated by other authorized documents.

AD No. _____
DDC FILE COPY

SPONSORED BY
THE ARMY MATHEMATICS STEERING COMMITTEE ON BEHALF OF

THE OFFICE OF
THE CHIEF OF RESEARCH, DEVELOPMENT AND
ACQUISITION

9) *Interim technical rept.*

U.S. ARMY RESEARCH OFFICE

14 ARB-

Report No. 77-3

11

November 1977

12) 6-15p.

6

PROCEEDINGS OF THE 1977 ARMY NUMERICAL
AND COMPUTERS ANALYSIS CONFERENCE,
Sponsored by the Army Mathematics Steering Committee

HOST

held at

Mathematics Research Center, University of Wisconsin,
Madison, Wisconsin
30-31 March 1977.

Approved for public release; distribution unlimited.
The findings in this report are not to be construed
as an official Department of the Army position, un-
less so designated by other authorized documents.

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, North Carolina

040900 ✓

4/5

FOREWORD

In a letter dated 24 February 1976, Professor J. Barkley Rosser, then Acting Director of the Mathematics Research Center (MRC), issued an invitation on behalf of the MRC Executive Committee to hold the 1977 Army Numerical Analysis and Computers Conference at the University of Wisconsin. He stated that the facilities of the Wisconsin Center had been reserved for a full week starting 28 March 1977. The first part of that week would be devoted to an MRC sponsored symposium on Mathematical Software, and during the rest of the week they would be happy to serve as the host of the Army conference. Holding these two meetings back-to-back was deemed an excellent idea, and the invitation issued by Dr. Rosser was readily accepted by the members of the Subcommittee on Numerical Analysis and Computers of the Army Mathematics Steering Committee (AMSC). The AMSC sponsors these conferences, and this subcommittee is responsible for their conduction. [For those interested in the names of the speakers at the Mathematical Software Symposium, the program of this meeting is printed on two of the following pages.]

The theme of the 1977 Army Numerical Analysis and Computers Conference was "The Numerical Solution of Partial Differential Equations". Many of the papers presented stressed ideas related to this theme. The keynote address was delivered by Professor Peter Lax of the Courant Institute of Mathematical Sciences. He spoke to the group on "Survey of Recent Techniques for Solving Hyperbolic Equations". The second hour address was entitled "Progress in the Calculation of Two and Three Dimensional Boundary Layers", and was delivered by Professor Tuncer Cebeci of California State University

Addresses delivered on this theme were

next page

at Long Beach. Drs. James Ortega and David A. Fisher gave the two other invited addresses. Dr. Ortega, ICASE, NASA at Langley Research Center, spoke on "Solution of Partial Differential Equations on Vector Computers". Dr. Robert G. Voight, also of ICASE, was a coauthor of this paper. David A. Fisher, of the Institute for Defense Analysis at Arlington, Virginia, chose to speak on the topic "Numeric Computation Facilities of a Common Programming Language for DOD". In addition to the invited addresses, there were several contributed papers. All of these talks were of high caliber, and we are pleased to announce that many of the papers presented at this meeting appear in these proceedings.

Members of the AMSC would like to thank MRC for serving as host of this conference; and to cite especially Professors Louis Rall and J. M. Yohe for the splendid way they handled the many problems associated with the local arrangements. Their thanks also go to all the speakers. They have asked that these proceedings be issued so that those who did attend this conference could have an opportunity to study the manuscripts of the presented papers, and other scientists not able to attend could still benefit from the scientific results obtained by the various authors.

ACCESSION for	
NTIS	Write Section <input checked="" type="checkbox"/>
DDC	Buy Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY STATEMENT	
DATE	
A	

SYMPOSIUM ON MATHEMATICAL SOFTWARE

PROGRAM

SUNDAY, MARCH 27, 1977

p.m.
8:00- Registration and Open House, Blue
10:00 Lounge, The Wisconsin Center, 702
 Langdon Street

MONDAY, MARCH 28, 1977

a.m.
8:00 Registration, first floor, The Wisconsin
 Center
8:45 Welcome, *Ben Noble*, Director, Mathe-
 matics Reach Center

SESSION I Chaired by *C. B. Moler*, University of
 New Mexico
9:00 Speaker: *G. H. Golub*, Stanford
 University
Topic: The Block Lanczos Method
 for Computing Eigenvalues
 (Joint work with *R. Underwood*,
 General Electric, San Jose)

10:00 Coffee, Exhibit Gallery
10:30 Speaker: *G. W. Stewart*, University of
 Maryland
Topic: Research, Development, and
 LINPACK

11:30 Speaker: *M. J. D. Powell*, Cambridge
 University
Topic: A Technique that Gains
 Speed and Accuracy in the
 Minimax Solution of Over-
 determined Linear Equations

p.m.
12:30 Lunch Break

SESSION II Chaired by *J. H. Griesmer*, IBM
 Research, Yorktown Heights
2:15 Speaker: *G. E. Collins*, University of
 Wisconsin-Madison
Topic: Infallible Calculation of Poly-
 nomial Zeros to Specified
 Precision

3:15 Coffee, Exhibit Gallery
3:30 Speaker: *R. E. Barnhill*, University of
 Utah
Topic: Representation and Approxi-
 mation of Surfaces

4:30 End of Session

TUESDAY, MARCH 29, 1977

a.m.
SESSION III Chaired by *W. J. Cody*, Argonne
 National Laboratory
9:00 Speaker: *C. W. Gear*, University of
 Illinois
Topic: Simulation: Conflicts
 Between Real Time and
 Software

10:00 Coffee, Exhibit Gallery
10:30 Speaker: *D. C. Hoaglin*, Harvard
 University
Topic: Mathematical Software and
 Exploratory Data Analysis

11:30 Speaker: *C. L. Lawson*, Jet Propul-
 sion Laboratory
Topic: Software for C¹ Surface
 Interpolation

p.m.
12:30 Lunch Break

SESSION IV Chaired by *W. Kahan*, University of
 California, Berkeley
2:15 Speaker: *W. R. Cowell*, Argonne
 National Laboratory/
L. D. Fosdick, University of
 Colorado
Topic: Mathematical Software
 Production

3:15 Coffee, Exhibit Gallery
3:30 Speaker: *W. S. Brown*, Bell
 Laboratories
Topic: Portability

4:30 End of Session
6:30 Cocktails (cash bar), Alumni Lounge,
 The Wisconsin Center, 702 Langdon
 Street
7:30 Dinner, The Wisconsin Center Dining
 Room

(This program is completed on the next page)

2.м.

9:00

Speaker: I. Babuška, University of Maryland

Topic: Computational Aspects of the Finite Element Method (joint work with W. Rheinboldt, University of Maryland)

10:30 Speaker: L. F. Shampine, Sandia Laboratories

Topic: The Art of Writing a Runge-Kutta Code

11:30 Speaker: A. Brandt, Weizmann Institute

Topic: Multi-Level Adaptive Techniques for Partial Differential Equations: Ideas and Software

12:30 **End of Program**

J. R. Rice, Chairman

J. R. Rice, Chairman

C. de Boor

J. M. Yohe

Gladys G. Moran, Secretary

TABLE OF CONTENTS*

Title	Page
Foreward	iii
Table of Contents	vii
Agenda	x
A Finite Element Method for Hyperbolic Systems of Equations in Two Spatial Dimensions and Time Applied to Unsteady Gas Flow James A. Schmitt	1
The Numerical Solution of Axisymmetric Free Boundary Porous Flow Well Problems Colin W. Cryer and Hans Fetter	27
Direct and Iterative One Dimensional Front Tracking Methods for the Two Dimensional Stefan Problem Gunter H. Meyer	39
On the Numerical Solution of the Poisson Equation by the Capacitance Matrix Method Arthur S. Shieh	53
A Power Series Solution of a Harmonic Mixed Boundary Value Problem J. Barkley Rosser and N. Papamichael	61
A Fourier-Solution of Parabolic PDE by Taylor Series Y.F. Chang	109
A "Sinc-Galerkin" Method of Solution of Boundary Value Problems Frank Stenger	123
On Boundary Extrapolation and Dissipative Schemes for Hyperbolic Problems Moshe Goldberg	157

* This Table of Contents lists only the papers that are published in this technical manual. For a list of all of the papers presented at the 1977 Army Numerical Analysis and Computers Conference see the copy of the Agenda.

ELLPACK: A Cooperative Effort for the Study of Numerical Methods for Elliptic Partial Differential Equations John R. Rice	165
Storage and Retrieval of Information on Systems of Partial Differential Equations and Their Solutions: Creatabase and the Continuum Mechanics Center Data Base of Hydrocodes Morton A. Hirschberg, Joseph Lacetera, James A. Schmitt . .	171
A Parallel Array Computer for the Solution of Field Problems W.R. Cyre, C.J. Davis, A.A. Frank, L. Jedynek, M.J. Redmond, V.C. Rideout	211
Software for Interval Arithmetic: A Reasonable Portable Package J.M. Yohe	237
Automatic Differentiation of Computer Programs Gershon Kedem	261
Progress in the Calculation of Two- and Three-Dimensional Boundary Layers Tuncer Cebeci	303
Approximation with VPB Splines Royce W. Soanes, Jr.	327
Best L_2 Approximation from Nonlinear Spline Manifolds I Unicity Results Charles K. Chui, Philip W. Smith, Jeff Chow	353
Best L_2 Approximation from Nonlinear Spline Manifolds II - Application to Optimal Quadrature Formula Charles K. Chui, Philip W. Smith, Joseph D. Ward	361
NL9 - An Adaptive Routine for Numerical Quadrature Arthur Hausner	367
On the Optimality of the Rayleigh-Ritz Approximation S.C. Eisenstat, R.S. Schreiber, M.H. Schultz	411
Numerical Solution of Gun Tube Problems in the Elastic- Plastic Range Peter C.T. Chen	423

Perturbation Methods for the Solution of Linear Problems L.B. Rall	441
Solution of Partial Differential Equations on Vector Computers James M. Ortega and Robert G. Voigt	475
A Mathematical Solution to the Ballistic Differential Equation Programmed for the Texas Instrument SR-52 Calculator Thomas H. Slook	527
Approximate Decompositions of Sparse Matrices Henk A. van der Vorst	539
Comments on the Solution of Coupled Stiff Differential Equations M.D. Kregel and J.M. Heimerl	553
A_0 -Stability of Brown's Multistep Multiderivative Methods Rolf Jeltsch	565
An Example of Apparent Convergence to the Wrong Limit Walter Gautschi	583
Float Point Computation Facilities for a Common Programming Language for the DOD A. Fisher	585
List of Attendees	597

A G E N D A

1977 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE Mathematics Research Center University of Wisconsin Madison, Wisconsin

All sessions will be held in the Wisconsin Center, Lake and Langdon Streets,
Madison, Wisconsin

Wednesday Afternoon, 30 March 1977

1300-1345 REGISTRATION - EXHIBIT GALLERY

1345-1400 OPENING REMARKS - AUDITORIUM

1400-1500 KEYNOTE ADDRESS

CHAIRPERSON - Hermann R. Robl, U. S. Army Research Office,
Research Triangle Park, North Carolina

SPEAKER - Peter Lax, Courant Institute of Mathematical
Sciences, New York, New York

TITLE - SURVEY OF RECENT TECHNIQUES FOR SOLVING HYPERBOLIC
EQUATIONS

1500-1515 BREAK - Coffee in the Exhibit Gallery

1515-1715 TECHNICAL SESSION I - AUDITORIUM

CHAIRPERSON - Norman Banks, Ballistic Research Laboratories,
Aberdeen Proving Ground, Maryland

A FINITE ELEMENT METHOD FOR SYSTEMS OF TIME DEPENDENT
HYPERBOLIC EQUATIONS IN TWO SPATIAL DIMENSIONS APPLIED
TO UNSTEADY GAS FLOW

James A. Schmitt, Ballistic Research Laboratories,
Aberdeen Proving Ground, Maryland

THE NUMERICAL SOLUTION OF POROUS FLOW FREE BOUNDARY
PROBLEMS

Colin W. Cryer, Mathematics Research Center, University
of Wisconsin, Madison, Wisconsin

LOCALLY ONE DIMENSIONAL METHODS FOR FREE BOUNDARY
PROBLEMS

Gunter H. Meyer, Georgia Institute of Technology,
Atlanta, Georgia

ON THE NUMERICAL SOLUTION OF POISSON'S EQUATION BY THE
CAPACITANCE MATRIX METHOD

Arthur Shieh, Mathematics Research Center, University
of Wisconsin, Madison, Wisconsin

A POWER SERIES SOLUTION OF A HARMONIC MIXED BOUNDARY
VALUE PROBLEM

J. Barkley Rosser and N. Papamichael, Mathematics
Research Center, University of Wisconsin, Madison,
Wisconsin

A FOURIER-SOLUTION OF PARABOLIC PDE BY TAYLOR SERIES

Y. F. Chang, University of Nebraska, Lincoln, Nebraska

A "SINC-GALERKIN" METHOD OF SOLUTION OF BOUNDARY VALUE
PROBLEMS

Frank Stenger, University of British Columbia, Vancouver,
B.C., Canada

1515-1715

TECHNICAL SESSION II - ROOM 210

CHAIRPERSON - Bruce Barnett, ARRADCOM, Dover,
New Jersey

A COOPERATIVE EFFORT FOR THE STUDY OF NUMERICAL METHODS
FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS-ELLPACK
John R. Rice, Purdue University, West Lafayette, Indiana

STOPAGE AND RETRIEVAL OF SYSTEMS OF PDES, THEIR SOLUTIONS,
AND IDENTIFYING INFORMATION

Morton A. Hirschberg*and Joseph Lacetera, Jr., Ballistic
Research Laboratories, Aberdeen Proving Ground, Maryland

A PARALLEL ARRAY COMPUTER FOR THE SOLUTION OF FIELD
PROBLEMS

W. R. Cyre, C. J. Davis, A. A. Frank, L. Jedynak,
M. J. Redmond, and V. C. Rideout*,University of
Wisconsin, Madison, Wisconsin

SOFTWARE FOR INTERVAL ARITHMETIC: A REASONABLY PORTABLE
PACKAGE

J. M. Yohe, Mathematics Research Center, University of
Wisconsin, Madison, Wisconsin

AUTOMATIC DIFFERENTIATION OF COMPUTER PROGRAMS

G. Kedem, Mathematics Research Center, University of
Wisconsin, Madison, Wisconsin

ON BOUNDARY EXTRAPOLATION AND DISSIPATIVE SCHEMES FOR
HYPERBOLIC PROBLEMS

Moshe Goldberg, University of California, Los Angeles,
California

1930-2100

PANEL DISCUSSION - LARGE SCALE MATHEMATICAL SOFTWARE
IN ARMY LABORATORIES - ROOM 316

PANEL MODERATOR - Paul T. Boggs, U. S. Army Research
Office, Research Triangle Park, North Carolina

PANEL MEMBERS - J. Michael Yohe, Mathematics Research
Center, University of Wisconsin, Madison, Wisconsin;
John Rice, Purdue University, West Lafayette, Indiana;
John Kring, Air Mobility R&D Laboratory, Cleveland,
Ohio; John Shipley, Air Mobility R&D Laboratory, NASA-
LANGLEY Research Center, Hampton, Virginia; Dennis
Tracey, U. S. Army Materials & Mechanics Research Center,
Watertown, Massachusetts; M. A. Hussain, Magqs Research
Center, Watervliet Arsenal, Watervliet, New York;
Norman Banks, Ballistic Research Laboratories, Aberdeen
Proving Ground, Maryland

Thursday, 31 March 1977

0830-0930

GENERAL SESSION I - AUDITORIUM

CHAIRPERSON - Robert E. Singleton, U. S. Army Research
Office, Research Triangle Park, North Carolina

SPEAKER - Tuncer Cebeci, California State University,
Long Beach, California

TITLE - PROGRESS IN THE CALCULATION OF TWO AND THREE
DIMENSIONAL BOUNDARY LAYERS

0930-1030

TECHNICAL SESSION III - AUDITORIUM

CHAIRPERSON - Dennis Tracey, U. S. Army Materials & Mechanics Research Center, Watertown, Massachusetts

APPROXIMATION WITH VFB-SPLINES

Royce W. Soanes, Jr., Watervliet Arsenal, Watervliet, New York

COMPUTATION OF L_2 SPLINE APPROXIMANTS WITH APPLICATIONS

Charles K. Chui², Philip W. Smith*, Jeff Chow; Texas A&M University, College Station, Texas

AN ADAPTIVE ROUTINE FOR NUMERICAL QUADRATURE

Arthur Hausner, Harry Diamond Laboratories, Adelphi, Maryland

0930-1030

TECHNICAL SESSION IV - ROOM 207

CHAIRPERSON - Edward Ross, U. S. Army Natick R&D Command, Natick, Massachusetts

ON THE OPTIMALITY OF THE RAYLEIGH-RITZ APPROXIMATION

S. C. Eisenstat, R. Schreiber*, M. H. Schultz, Yale University, New Haven, Connecticut

THE APPROXIMATE SOLUTION OF GENERALIZED HAMILTONIAN PROBLEMS

B. Noble* and J. V. Zago, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin

GALERKIN METHODS FOR ROTATIONALLY SYMMETRIC PARTIAL DIFFERENTIAL EQUATIONS

Dennis C. Jespersion, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin

0930-1030

TECHNICAL SESSION V - ROOM 313

CHAIRPERSON - Sylvan Eisman, Frankford Arsenal, Philadelphia, Pennsylvania

NUMERICAL SOLUTION OF GUN TUBE PROBLEMS IN THE ELASTIC-PLASTIC RANGE

Peter C. T. Chen, Watervliet Arsenal, Watervliet, New York

REMARKS ON NUMERICAL ANALYSIS OF RADON INTEGRAL EQUATION AND PICTURE RECONSTRUCTION FROM PROJECTIONS

M. Z. Nashed, University of Michigan, Ann Arbor, Michigan

PERTURBATION METHODS FOR THE SOLUTION OF LINEAR PROBLEMS

L. B. Rall, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin

1030-1045

BREAK - Coffee in the Exhibit Gallery

1045-1145

GENERAL SESSION II - AUDITORIUM

CHAIRPERSON - M. A. Hussain - Watervliet Arsenal, Watervliet, New York

SPEAKER - Dr. James M. Ortega, ICASE, Nasa Langley Research Center, Hampton, Virginia

TITLE - SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS ON VECTOR COMPUTERS (Dr. Robert G. Voigt, Co-author)

1145-1300	LUNCH
1300-1400	<p>TECHNICAL SESSION VI - AUDITORIUM</p> <p>CHAIRPERSON - Shirley J. Smith, Aviation Systems Command, St. Louis, Missouri</p> <p>A MATHEMATICAL SOLUTION TO THE BALLISTIC DIFFERENTIAL EQUATION PROGRAMMED FOR THE TEXAS INSTRUMENT SR-52 T. H. Slook, Temple University and Frankford Arsenal, Philadelphia, Pennsylvania</p> <p>APPROXIMATE DECOMPOSITIONS OF SPARSE MATRICES H. A. van der Vorst, European Research Office, London, and Academic Computer Center Utrecht, Netherlands</p> <p>A GLOBAL ELEMENT METHOD FOR ELLIPTIC P.D.E.'s L. M. Delves, University of Victoria, Victoria, British Columbia, Canada</p>
1300-1400	<p>TECHNICAL SESSION VII - ROOM 227</p> <p>CHAIRPERSON - Peter C. T. Chen, Watervliet Arsenal, Watervliet, New York</p> <p>COMMENTS ON THE SOLUTION OF COUPLED STIFF DIFFERENTIAL EQUATIONS M. D. Kregel and J. M. Heimerl, Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland</p> <p>A - STABILITY OF BROWN'S MULTISTEP MULTIDERIVATIVE METHODS Rolf Jeltsch, Ruhr University Bochum, Germany</p> <p>AN EXAMPLE OF APPARENT CONVERGENCE TO THE WRONG LIMIT Walter Gautschi, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin</p>
1400-1415	BREAK - Coffee in the Exhibit Gallery
1415-1515	<p>GENERAL SESSION III - AUDITORIUM</p> <p>CHAIRPERSON - R. P. Uhlig, U. S. Army Materiel Development and Readiness Command, Alexandria, Virginia.</p> <p>SPEAKER - David A. Fisher, Institute for Defense Analysis, Arlington, Virginia</p> <p>TITLE - NUMERIC COMPUTATION FACILITIES OF A COMMON PROGRAMMING LANGUAGE FOR THE DOD</p>
1515-1615	OPEN MEETING OF NUMERICAL ANALYSIS SUBCOMMITTEE - AUDITORIUM

A FINITE ELEMENT METHOD FOR HYPERBOLIC SYSTEMS OF
EQUATIONS IN TWO SPATIAL DIMENSIONS AND TIME
APPLIED TO UNSTEADY GAS FLOW

James A. Schmitt
U.S. Army Ballistic Research Laboratory
Ballistic Modeling Division
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. The report contains a discussion of a numerical method for solving systems of first order time dependent hyperbolic equations in two spatial variables. This scheme which combines the finite element methodology and the properties of a hyperbolic system of differential equations is applied to unsteady gas flow problems. The formulation is based on the elementwise least squares minimization of the differential residual error and on the construction of the finite elements in both space and time. The methodology is presented in detail. Numerical experiments involving both smooth and shocked flows are discussed. Areas of possible future code development are proposed.

1. INTRODUCTION. Although the finite element method is a proven, effective method for obtaining numerical solutions of solid mechanics problems [1]¹, its impact on computational fluid dynamics has been felt only in the past few years [2]. Because of the diverse applications of this method in continuum mechanics, many departures from the original method used in structural analysis have been made. Our adaptation of the finite element method for direct application to unsteady gas flows in two spatial variables is based, in part, on Lynn and Arya's least squares formulation [3,4] and on Polk's one dimensional study [5]. Lynn and Arya's approach is based on the elementwise least squares minimization of the differential residual error which allows a direct finite element formulation from the governing differential equations. Furthermore, since the governing equations are hyperbolic, the finite elements can be constructed in both space and time so that they approximate the domain of determinacy associated with hyperbolic problems. Polk combined these two concepts and applied them to the unsteady isentropic flow of an inviscid gas expanding behind a piston. Using both linear and quadratic approximations to the dependent variables, he showed good agreement between the finite element results and the exact solution for the smooth portion of the flow. Near the gradient discontinuities which occurred in the flow, Polk constructed the finite elements so that a side of the element and the locus of discontinuities coincided. Using the special construction, good agreement was obtained everywhere in the flow.

This report is concerned with a finite element method for unsteady inviscid compressible flows in two spatial dimensions, a corresponding pilot computer code and some resulting numerical experiments. The system of

¹ Numbers in brackets designate references at end of report.

governing equations are nonlinear hyperbolic equations. We take advantage of this fact to simplify the general finite element method. This is contrary to Wellford and Oden's [6] parabolic regularization method for hyperbolic problems. In Wellford and Oden's technique, certain terms which depend on the discretization parameters are appended to the equations so that they become parabolic. This parabolic problem is then solved by a finite element technique. It can be shown for a class of problems that the solution of the parabolic problem converges to the original hyperbolic solution in the limit as the mesh size tends to zero. On the other hand, our formulation deals directly with the hyperbolic equations.

Our construction of the finite elements is reminiscent of Polk's construction in that they are in both space and time but differ in that they enclose, not coincide with, the domain of dependence. It will be shown that the construction simplifies the necessary integration routines while satisfying a Courant condition. No special construction of the finite element is provided near steep gradients and discontinuities in the flow. Consequently, no special knowledge of the solution is required and all interior nodes in the calculation are treated identically. Furthermore, by applying Lynn and Arya's least squares minimization to each finite element, we can avoid the large matrices generally associated with the finite element method for elliptic problems while still retaining the essential advantages of the method.

The general methodology (the construction of the finite elements, the approximations to the flow variables and the formulation in terms of the least squares error criterion) is explained in section 2. Section 3 contains a brief discussion of the form of the governing equations and certain approximations used within the code. The results of two numerical experiments are given and discussed in section 4. Section 5 contains a summary of the method and areas in which future work is required.

2. GENERAL METHODOLOGY. The first step in the finite element methodology is to divide the solution region into elements. We divide the computational domain for a given time (a two dimensional region) into triangular elements. The vertices of the triangles are called nodes. We assume that the boundaries are stationary so that the triangular divisions remain unchanged with time. The system of governing equations for compressible fluid flow include coupled nonlinear partial differential equations which express conservation of mass, momentum and energy plus an algebraic equation of state. Because the governing equations are hyperbolic, the solution at a point $(x, y, \bar{t} + \Delta t)$ in the solution domain depends only on the value of the flow variables within the intersection of the domain of dependence (the mach cone) from the point $(x, y, \bar{t} + \Delta t)$ and the (x, y, \bar{t}) plane. Consequently, given a union of triangles in the (x, y, \bar{t}) plane with a vertex at the point $(\bar{x}, \bar{y}, \bar{t})$ and the values of the flow variables at the corresponding nodes, we can compute a value of Δt such that the mach cone from the point $(\bar{x}, \bar{y}, \bar{t} + \Delta t)$ lies within the union of triangles. Since the computed value of Δt will vary from node to node, we take the minimum value of Δt over all nodes as the next time step. This value allows a systematical advance in time.

We now define our finite element at a point $(\bar{x}, \bar{y}, \bar{t} + \Delta t)$ as the union of all prisms with a base vertex at the point (x, y, \bar{t}) and with a uniform height Δt . See Figure 1. The present one offers several desirable simplifications over other possible constructions of the finite element. From the discussion above, it is clear that the values of the flow variables at a node are independent of the values at the other nodes at the same time level. Thus, the interconnection of the nodal values which is characteristic of finite element formulations of elliptic problems and which result in the manipulation of large matrices can and will be avoided. We will solve for the central nodal values at this new time level by considering only the finite element at the central node. Furthermore, any necessary time integrations over the finite element are simplified, since the nodes of the elements are independent of time.

The next step is to choose the interpolating or trial functions over the finite element. Let ω be a flow variable; that is, a dependent variable computed directly from the governing differential equations, not the equation of state. The interpolating function for ω is

$$\omega(x, y, t) = \omega_0(x, y, \bar{t}) + t \cdot (a_i x + a_{i+1} y + a_{i+2}), \quad (1)$$

where the points (x, y, t) lie within the finite element at $(\bar{x}, \bar{y}, \bar{t} + \Delta t)$ and the parameters a_i, a_{i+1}, a_{i+2} are to be determined. The function $\omega_0(x, y, \bar{t})$ represents the flow variable ω at time \bar{t} and is assumed known. These interpolation functions form over each finite element a linear approximation in space to the time derivative of ω .

To complete the model, a technique to determine the parameters a_i and thus the flow variables, is needed. A basic idea in the finite element methodology is to minimize the errors occurring from the residual of the governing differential equations in terms of the interpolating functions. Thus, the finite element method approximates the minimum residual whereas the finite difference method approximates the differential equations. Since the proposed analysis is elementwise, we desire a minimization technique for each element. To this end, we choose the elementwise least squares minimization of the differential residual error employed by Lynn and Arya.

We substitute the interpolating functions into the k -th governing differential equation, make the result dimensionless and denote the resulting residual by $D_k(x, y, t; \vec{a})$, where \vec{a} is the vector of unknown parameters a_i . Each residual D_k is dimensionless so that no individual D_k will numerically dominate the least squares sum of all the residuals because of dimensional disparities. The D_k 's are explicit algebraic functions of the independent variables x, y, t and the parameters \vec{a} . The total error in the sense of least squares over a particular finite element is denoted by $E(\vec{a})$ and is given by

$$E(\vec{a}) = \iiint_V \sum_{k=1}^{NOEQ} D_k^2(x, y, t; \vec{a}) \, dx dy dt, \quad (2)$$

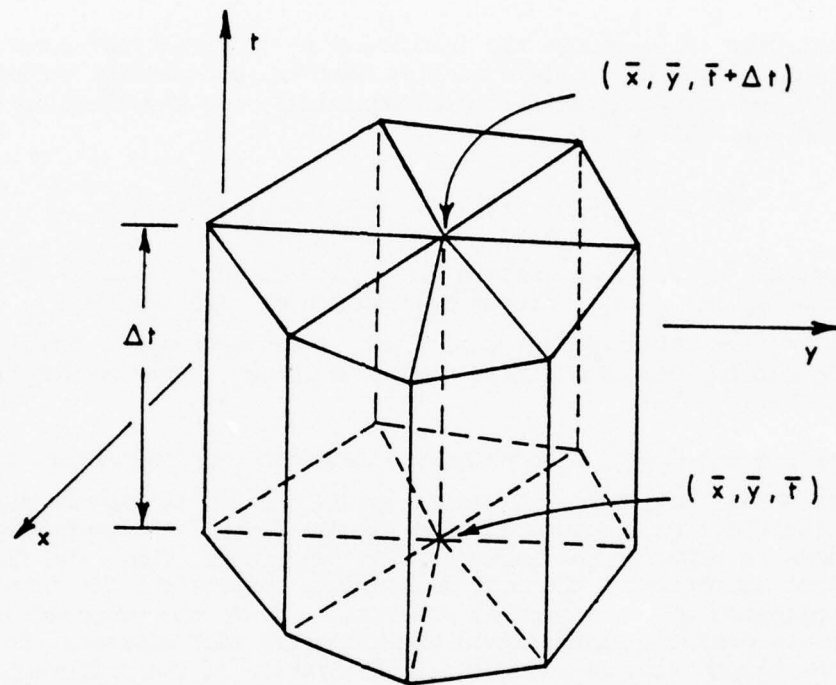


Figure 1. Typical Finite Element at an Interior Node $(\bar{x}, \bar{y}, \bar{t} + \Delta t)$.

where NOEQ is the number of governing equations and V is the volume of the finite element. We wish to minimize $E(\bar{a})$ with respect to the a_i 's. A necessary condition for the existence of a minimum is $\partial E / \partial a_i = 0$ or

$$\iiint_V \sum_{k=1}^{NOEQ} D_k \frac{\partial D_k}{\partial a_i} dx dy dt = 0, \text{ for each } i. \quad (3)$$

Note that the derivatives $\partial D_k / \partial a_i$ can be explicitly calculated. By solving this nonlinear algebraic system of equations for the unknowns a_i , we can determine the values of the flow variables at the nodal point $(\bar{x}, \bar{y}, \bar{t} + \Delta t)$. We repeat this process for each interior node in the solution domain.

For a boundary node, the above procedure is slightly altered. We rewrite the given boundary condition(s) at the boundary node in terms of the interpolating functions (1). We then solve for the unknown parameters a_i in equation (3) at the boundary node subject to the rewritten boundary conditions. Thus, at a boundary node we no longer have a pure minimization problem as at an interior node, but rather a constrained minimization problem.

3. PILOT COMPUTER PROGRAM. The pilot computer code (see Schmitt [7] for the program listing) is written in a modular structure fashion in order to clarify the logic of the program and to allow changes in the form of the governing equations, the interpolating functions, the method selected to solve the nonlinear system, etc.. Such flexibility is highly desirable for this pilot code. For example, in the finite difference techniques, the formulation of the equations have a profound affect on a method's performance (see, for example, Moretti [8]). Similarly, the form of the equations may affect the performance of the finite element method.

The code has three major components. The first component accepts the geometric and control parameters. Furthermore, this section accepts and/or generates the nodal positions within the x-y solution subdomain and necessary initial and boundary value data. The second component calculates the time increment and the flow variables' values at each node at the new time. The "heart" of the pilot code is clearly the second component, since the general method outlined in section 2 is implemented in this portion of the code. The third component provides the output and graphics capabilities for the program. The code uses the non-conservation inviscid form of the governing equations in Cartesian coordinates, where body forces, heat absorption and heat fluxes are neglected, the Newton-Raphson iteration method and certain approximations. We briefly discuss this specific situation below.

The governing equations in dimensional variables are:

$$\frac{\partial \rho}{\partial t} + u \frac{\partial \rho}{\partial x} + v \frac{\partial \rho}{\partial y} + \rho \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0, \quad (4)$$

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) + \frac{\partial p}{\partial x} = 0, \quad (5)$$

$$\rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) + \frac{\partial p}{\partial y} = 0, \quad (6)$$

$$\rho \left(\frac{\partial e}{\partial t} + u \frac{\partial e}{\partial x} + v \frac{\partial e}{\partial y} \right) + \rho \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) e = 0, \quad (7)$$

$$\rho = \rho(p, e). \quad (8)$$

Here the spatial coordinates are x and y , t is the time, u is the x -component of the velocity, v is the y -component of the velocity, p is the pressure, ρ is the density and e is the internal energy per unit mass. The functional form of the equation of state is given by equation (8). The particular equation of state used in a given calculation is specified in one of the program's subroutines.

In the actual calculations for a given finite element at the point $(\bar{x}, \bar{y}, \bar{t} + \Delta t)$ we translate the origin of the coordinate system to the point $(\bar{x}, \bar{y}, \bar{t})$. Conceptually, the translation enables each finite element to have its base centered at the same point $(x, y, t) = (0, 0, 0)$ and practically, it simplifies the calculations. The interpolating functions for the variables u, v, p, e within a finite element are:

$$u(x, y, t) = u_0(x, y, 0) + t \cdot (a_1 x + a_2 y + a_3), \quad (9)$$

$$v(x, y, t) = v_0(x, y, 0) + t \cdot (a_4 x + a_5 y + a_6), \quad (10)$$

$$p(x, y, t) = p_0(x, y, 0) + t \cdot (a_7 x + a_8 y + a_9), \quad (11)$$

$$e(x, y, t) = e_0(x, y, 0) + t \cdot (a_{10} x + a_{11} y + a_{12}), \quad (12)$$

where the variables x, y, t are in the translated finite element, the variables with subscript zero denote the variables at the known time level $t = 0$, and the a_i 's, $i = 1, 2, \dots, 12$ are the unknown parameters. We define the dimensionless residuals $D_k(x, y, t; \vec{a})$, $k = 1, 2, 3, 4$ as the quantities obtained by substituting equations (8) - (12) into the four differential equations (4) - (7), respectively, and then by dividing the first result by a ratio of a characteristic density to Δt , the second and third results by a ratio of a characteristic velocity to Δt and the fourth by a ratio of a characteristic internal energy per mass to Δt .

The values of the characteristic quantities are the density, sound speed and internal energy per unit mass at the center node and $\Delta t = 0$. The derivatives of ρ with respect to the variables x , y and t can be obtained by the chain rule. The subsequent partial derivatives of ρ with respect to p and e are calculated directly from the equation of state and are coded into the sub-routine associated with the equation of state. The functions u_0 , v_0 , p_0 , e_0 are estimated by a linear approximation on each triangle based on their values at the vertices. Although other approximations are possible, the present one is easily computable and is independent of the number of prisms in the finite element. The partial derivatives of u_0 , v_0 , p_0 , e_0 , required in $D_k(x, y, t; \vec{a})$ vary from triangle to triangle and consequently equation (3) must be rewritten as

$$F_i(\vec{a}) = \frac{\partial E(\vec{a})}{\partial a_i} = \sum_{\ell=1}^{\text{NUMTRI}} \int_{\ell} \int_{\text{prism}} \sum_{k=1}^4 D_k \frac{\partial D_k}{\partial a_i} dx dy dt = 0, \quad i=1,2,\dots,12, \quad (13)$$

where NOEQ is now four and the finite element contains NUMTRI prisms (or triangles).

The Newton-Raphson method is used to solve the system of nonlinear algebraic equations (13). Since the terms $D_k(x, y, t; \vec{a})$ are known functions of the parameter \vec{a} , the second partial derivatives of the least squares error can be explicitly calculated and are given by

$$\begin{aligned} \frac{\partial F_i(\vec{a})}{\partial a_j} = \frac{\partial}{\partial a_j} \left(\frac{\partial E(\vec{a})}{\partial a_i} \right) = \sum_{\ell=1}^{\text{NUMTRI}} \left\{ \sum_{k=1}^4 \int_{\ell} \int_{\text{prism}} \left[D_k \frac{\partial^2 D_k}{\partial a_i \partial a_j} \right. \right. \\ \left. \left. + \frac{\partial D_k}{\partial a_i} \frac{\partial D_k}{\partial a_j} \right] dx dy dt \right\}, \quad i,j=1,2,\dots,12. \end{aligned} \quad (14)$$

The initial values for a_3, a_6, a_9, a_{12} in the Newton-Raphson scheme for the finite element at $(0, 0, \Delta t)$ are taken as the previously computed values of these parameters for the finite element at $(0, 0, 0)$. However, at the first time step a special calculation is needed to find the appropriate initial values. These parameters are determined by solving for $\partial/\partial t$ in each of the equations (4) - (7), by approximating the spatial dependence of the variables at the initial time by linear functions on each triangle, by finding the corresponding values of $\partial/\partial t$ at each vertex, by equating these to the time

partials of equations (9) - (12), by averaging the resulting values of the a_i 's over the triangles and by translating the result. In all cases, the initial values of $a_1, a_2, a_4, a_5, a_7, a_8, a_{10}$ and a_{11} are set to zero. In the sample problems, the iterations converged faster with the zero values than with the more obvious choice $a_i \neq 0, i \neq 3, 6, 9, 12$. The integrals of $D_k \cdot (\partial^2 D_k / \partial a_i \partial a_j)$, $D_k \cdot (\partial D_k / \partial a_i)$ and $(\partial D_k / \partial a_i) \cdot (\partial D_k / \partial a_j)$ are approximated by a two point Gaussian quadrature in time and by a product of first order functions in space. For example,

$$\iiint_{\text{prism}} \frac{\partial D_k}{\partial a_i} \frac{\partial D_k}{\partial a_j} dx dy dt \approx \frac{\Delta t}{2} \sum_{m=1}^2 \left[\iint_{\text{triangle}} \frac{\partial D_k}{\partial a_i} (x, y, t_m; \vec{a}) \right. \\ \left. \times \frac{\partial D_k}{\partial a_j} (x, y, t_m; \vec{a}) dx dy \right], \quad (15)$$

where $t_m, m = 1, 2$, are the Gaussian quadrature points between zero and Δt .

In the spatial integrals at each Gaussian time, both factors are approximated by first order functions in x and y which are then multiplied and integrated exactly over the desired triangle. Once the values of the parameters a_i are known, the desired values of the flow variables at the center node at the new time ($t = \Delta t$) can be determined easily from equations (9) - (12).

The above discussion applies equally to interior and boundary type nodes. However, as noted at the end of section 2, the minimization at a boundary node is a constrained minimization. The type of constraints will depend on the type of boundary condition imposed at a given node. As an example, the zero normal velocity boundary condition imposed at a solid wall is incorporated into the equation system (13) in Appendix A.

4. NUMERICAL EXPERIMENTS. In this section the results of two non-steady flow calculations, the flow behind a cylindrical blast wave and the flow across a propagating normal shock, are presented. These two examples are computed in order to ascertain the scheme's characteristics on interior nodes for a smooth and discontinuous flow, respectively. Both of these time-dependent flows are essentially one dimensional in space; however, they will be treated as two dimensional problems. Furthermore, since closed-form solutions are known for each problem, the accuracy of the finite element formulation can be precisely evaluated.

The similarity solution of the blast wave problem is discussed by Sedov [9]. The flow behind a cylindrical blast wave rather than a planar wave is computed because of its associated circular solution domain. The computational domain for a given time consists of an annulus from radius r_a to radius r_b which is the position of the shock front at the initial time t_0 . Perhaps the most important advantage of the finite element method is its capability of dealing with complex geometrical shapes by using arbitrarily shaped simple elements. The ease with which this cylindrical problem is handled by the Cartesian finite element program, especially the circular boundaries, demonstrates this advantage in an elementary manner. In Figure 2, the first quadrant of a computation domain for a given time is drawn, where $r_a = 2.2$ [m], $r_b = 3.0$ [m], the nodes are equally spaced at four degree intervals on a given circle and the radial divisions are computed so that approximate equilateral triangles result. Consequently, the size of the triangles increase with the radial distance from the origin. Note the good approximation to the arc boundaries by the series of straight lines forming the base of the triangles. A finite difference method in Cartesian coordinates either would approximate the arc boundaries by a series of horizontal and vertical lines or would require a transform of the solution domain. In both cases special treatment of the boundaries would be required. On the other hand, no special programming is needed to treat the arc in the finite element code.

Numerical calculations were performed for a cylindrical blast wave [9, pp. 219-20] generated by the instantaneous release of a finite amount of energy proportional to $E_0 = 6.887025656 \times 10^6$ [J/m] into a gas with initial density $\rho_0 = 1.262523446$ [kg/m³]. For the calculations, the specific heat ratio of the perfect gas is $\gamma = 1.4$, the radii are $r_a = 2.2$ [m], $r_b = 3.0$ [m], and the initial time is $t_0 = 3.85342034 \times 10^{-3}$ [s]. On the initial time plane the values of flow variables are calculated from the exact solution. For the inflow conditions at r_a and outflow condition at r_b we again assign the exact values to the flow variables. The computed ratios of p/p_s , v_r/v_r , e/e_s and ρ/ρ_s (subscript s denotes value at the shock front) are compared to Sedov's exact values (solid line) in Figures 3, 4, 5 and 6, respectively. The symbols \triangleright and \times denote the computed value on the triangular finite element mesh with nodes spaced at two degree intervals (the computed radial subdivisions range from 0.07 [m] to 0.09 [m]) and at four degree intervals (the computed radial subdivisions range from 0.14 [m] to 0.18 [m]), respectively. Both sets of values are at the same time ($4.12516608 \times 10^{-3}$ [s]). The former used four timesteps ($\Delta t \approx 0.68 \times 10^{-4}$ [s]) and the latter two timesteps ($\Delta t \approx 1.36 \times 10^{-4}$ [s]) to reach the termination time. The maximum absolute value of the percent relative error for the ratios of pressure, radial velocity, internal energy and density are 1.15%, 0.12%, 0.10% and 1.24%, respectively, for the finer mesh and 2.68%, 0.30%, 0.36% and 3.04%, respectively, for the coarser mesh. We recall that the density is computed from the equation of state once

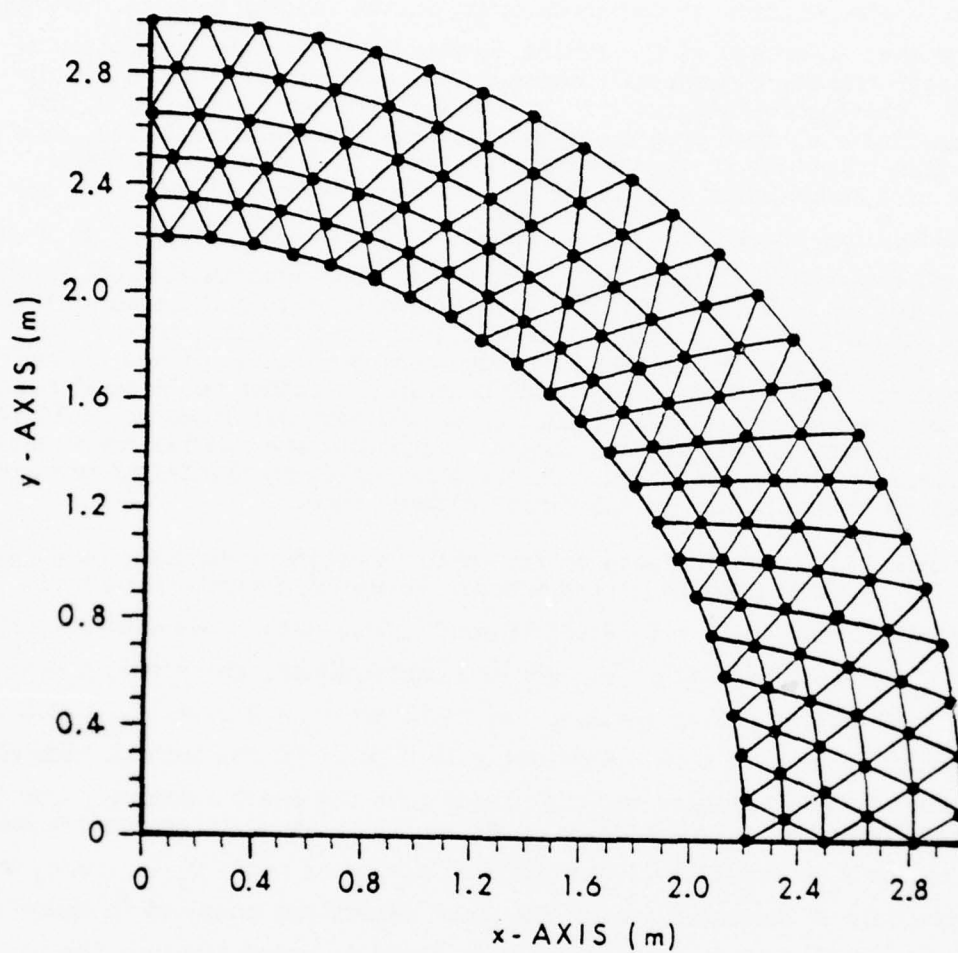


Figure 2. First Quadrant of Finite Element Mesh for Cylindrical Blast Wave Problem with Nodes at Four Degree Intervals.

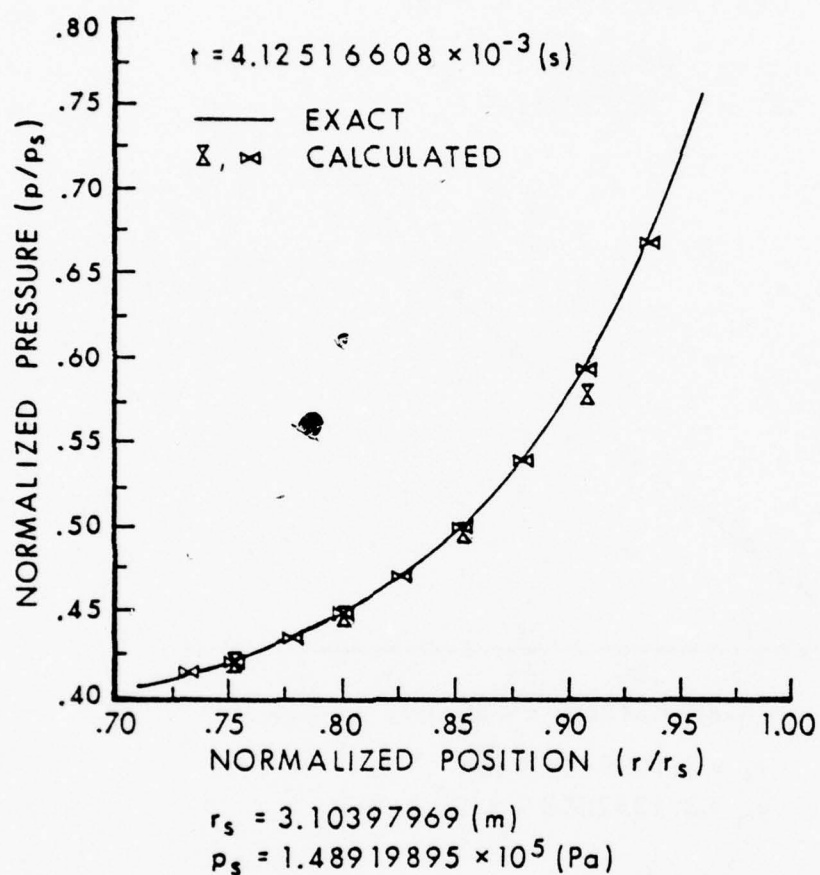


Figure 3. Comparison of Exact Normalized Pressure for Cylindrical Blast Wave [9] with Computed Values for Nodes Spaced at 4° (X) and 2° (X) Intervals.

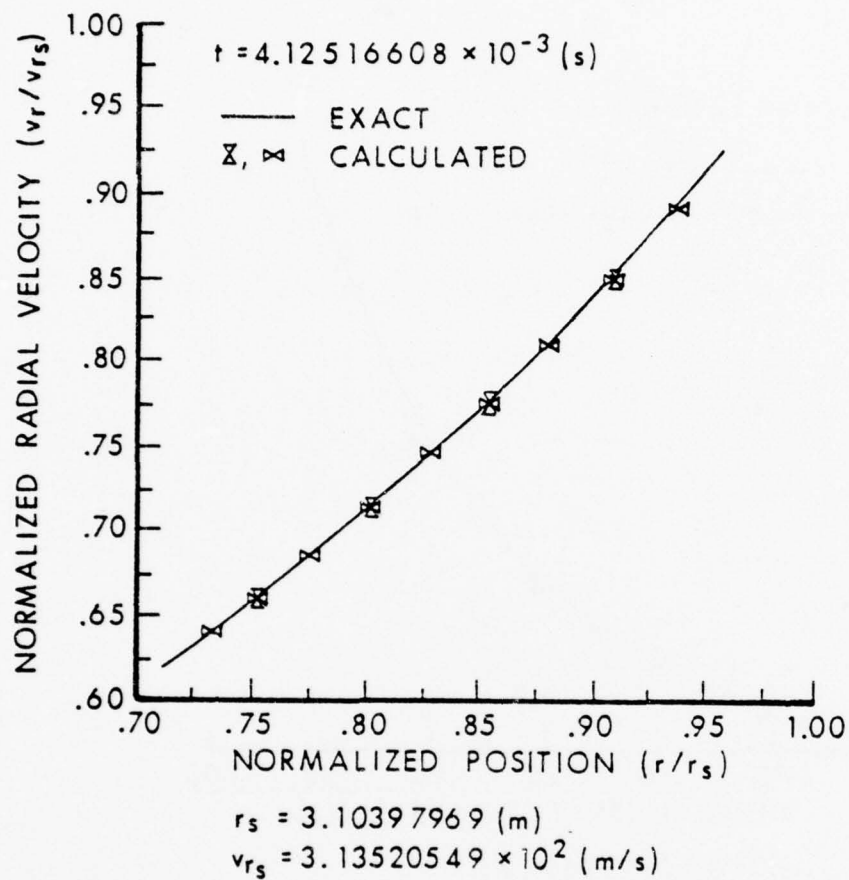


Figure 4. Comparison of Exact Normalized Radial Velocity for Cylindrical Blast Wave [9] with Computed Values for Nodes Spaced at 4° (X) and 2° (x) Intervals.

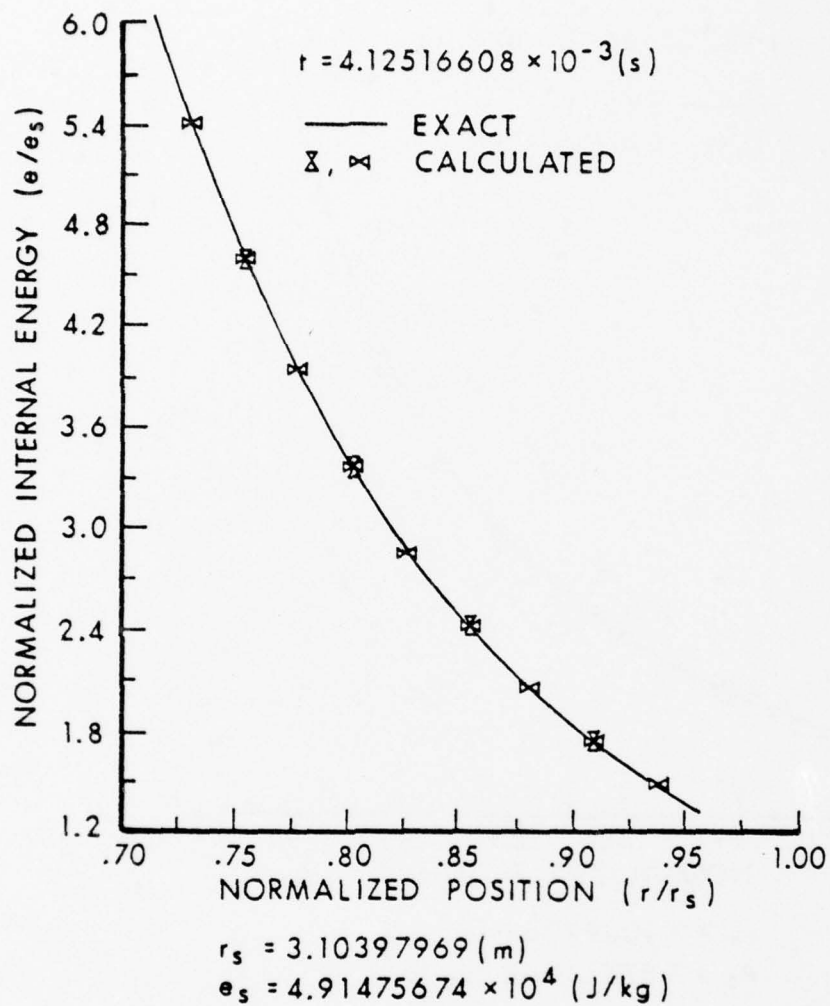


Figure 5. Comparison of Exact Normalized Internal Energy for Cylindrical Blast Wave [9] with Computed Values for Nodes Spaced at 4° (\times) and 2° (\diamond) Intervals.

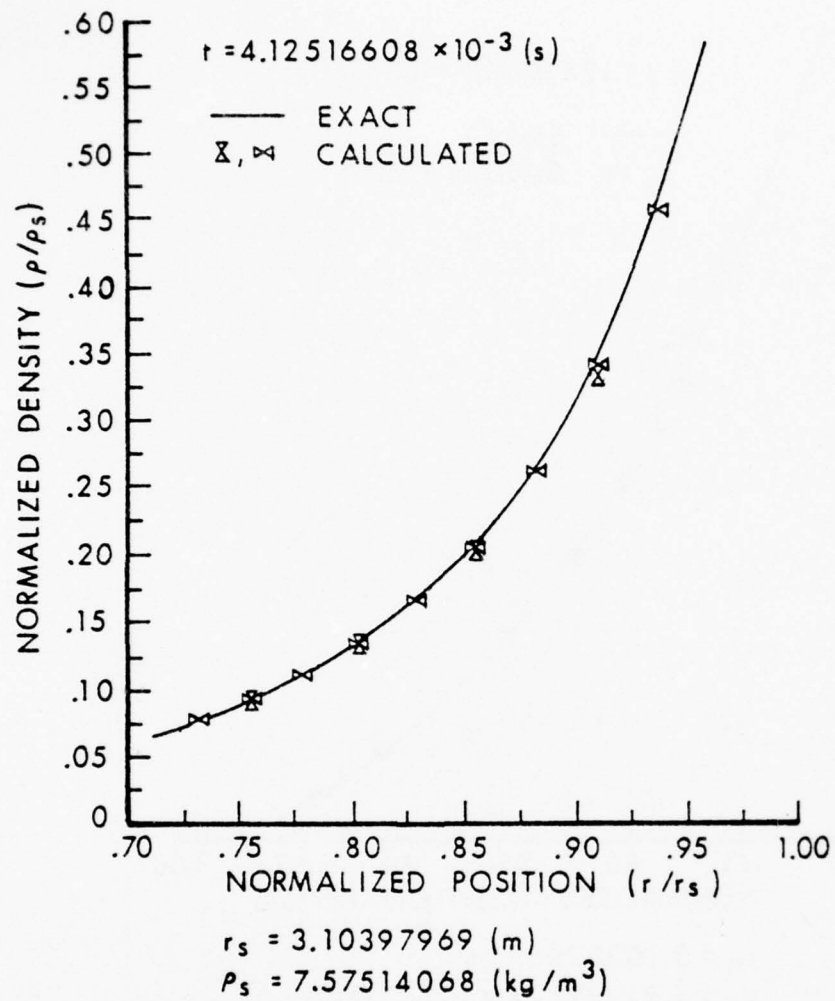


Figure 6. Comparison of Exact Normalized Density for Cylindrical Blast Wave [9] with Computed Values for Nodes Spaced at 4° (\times) and 2° ($>$) Intervals.

the error in the density includes both the pressure and energy errors. We note that the maximum relative error in each flow variable occurs where the finite elements are the largest; that is, near r_b . At the opposite end, near r_a , the finite elements are the smallest and the absolute value of the percent relative error of the pressure, radial velocity, internal energy and density ratios for the finer and coarser mesh are 0.11%, 0.01%, 0.02%, 0.14% and 0.44%, 0.18%, 0.19%, 0.64%, respectively. Hence, overall and despite the relatively large size of the triangular elements, the agreement is fairly good.

A normal shock of strength five ($p_2/p_1 = 5$) propagating into a rectangular field is the second calculation. The subscripts 1 and 2 on the variables denote their value in the pre- and post-shock states, respectively. The flow domain consists ideally of the infinite slab of the (x, y) plane, $-\infty < x < \infty$, $y_2 \leq y \leq y_1$ for time greater than the initial time t_0 . The shock is initially at the position y_{s_0} , $y_2 < y_{s_0} < y_1$, and is moving in the positive y -direction. The quiescent state is characterized by zero velocities ($u_1 = v_1 = 0$ [m/s]) and by pressure ($p_1 = 1.01325 \times 10^6$ [Pa]) and density ($\rho_1 = 1.225570786$ [kg/m³]) of air (specific heat ratio $\gamma = 1.4$) at sea level and at temperature 288.15° [K]. The values of the flow variables in the disturbed region are computed by the Rankine Hugoniot relations [10 pp. 62-65]. The nodes are equally spaced along constant y values and the y directed divisions are computed so that approximate equilateral triangles result. The computational domain for a given time is very similar to that in Figure 2 except that the radial and angular variables now correspond to the y and x variables, respectively. However, all the triangles in this case are the same size. On the initial time plane t_0 , the values of the flow variables are calculated from the exact solution. The boundaries at y_2 and y_1 are taken sufficiently far away from the shock front so that the constant values in both the disturbed and quiescent regions are obtained by the flow variables.

For the calculations, the y directed subdivisions are small in order to model the shock with its extremely thin thickness. The y directed divisions are 0.005 [m], $y_2 = 2.965$ [m], $y_1 = 3.050$ [m], $y_{s_0} = 3.0025$ [m] and

$t_0 = 0.0$ [s]. The graphs of the computed ratios for the pressure (p/p_s), the y -velocity (v/v_s), the internal energy (e/e_s) and the density (ρ/ρ_s) versus the y -axis are given in Figures 7, 8, 9, 10, respectively, for three times; 1.1524902×10^{-4} [s] (fourth time level), 2.5986071×10^{-4} [s] (ninth time level) and 4.0654310×10^{-4} [s] (fourteenth time level). The symbol \propto denotes the theoretical position of the shock front. The Figures 7-10 show fair resolution of the locations of this shock front for the given times. Spurious oscillations develop at the shock front as expected, since no artifice was introduced in the equations or scheme to suppress them. The oscillations occur about the exact solutions $p/p_s = v/v_s = e/e_s = \rho/\rho_s = 1$ in the disturbed flow

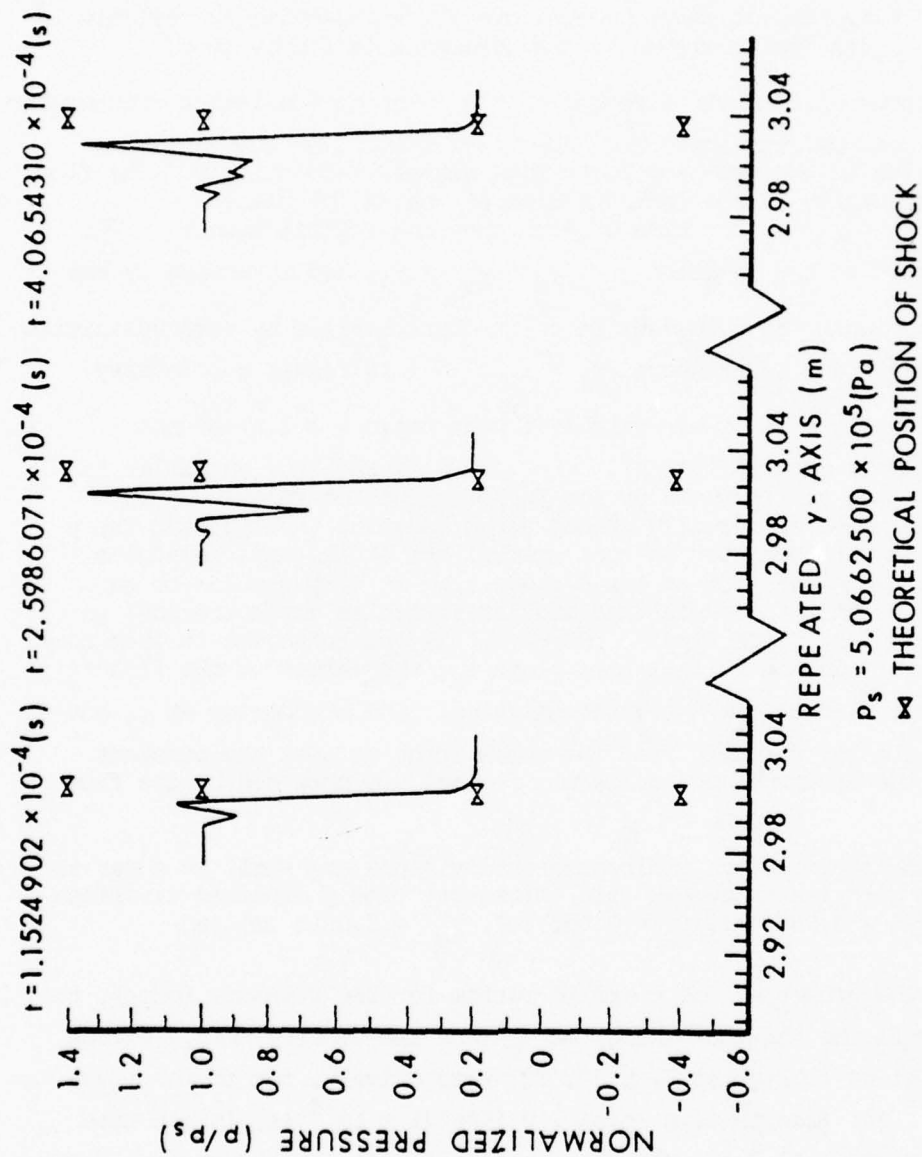


Figure 7. Variation of Normalized Pressure with Distance for Normal Shock of Strength Five at Three Times.

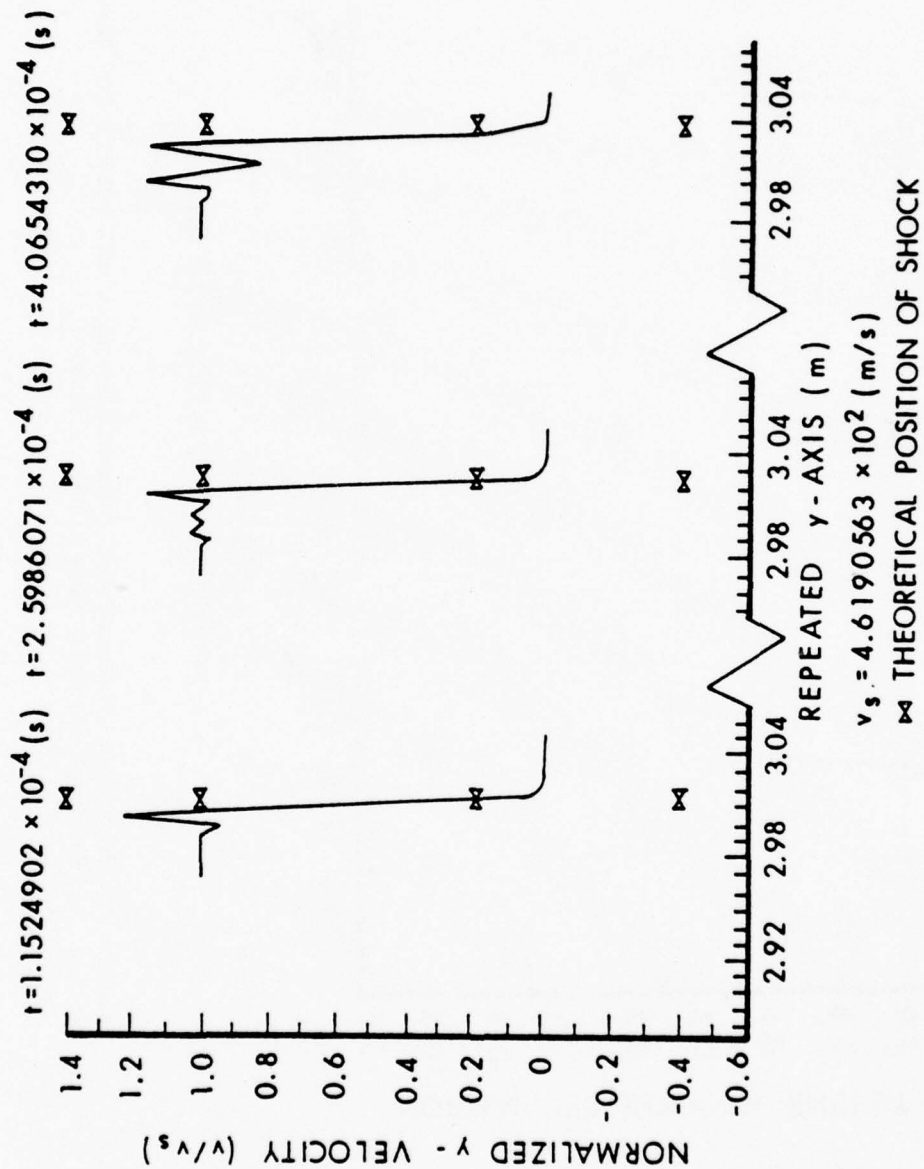


Figure 8. Variation of Normalized y Component of Velocity with Distance for Normal Shock of Strength Five at Three Times.

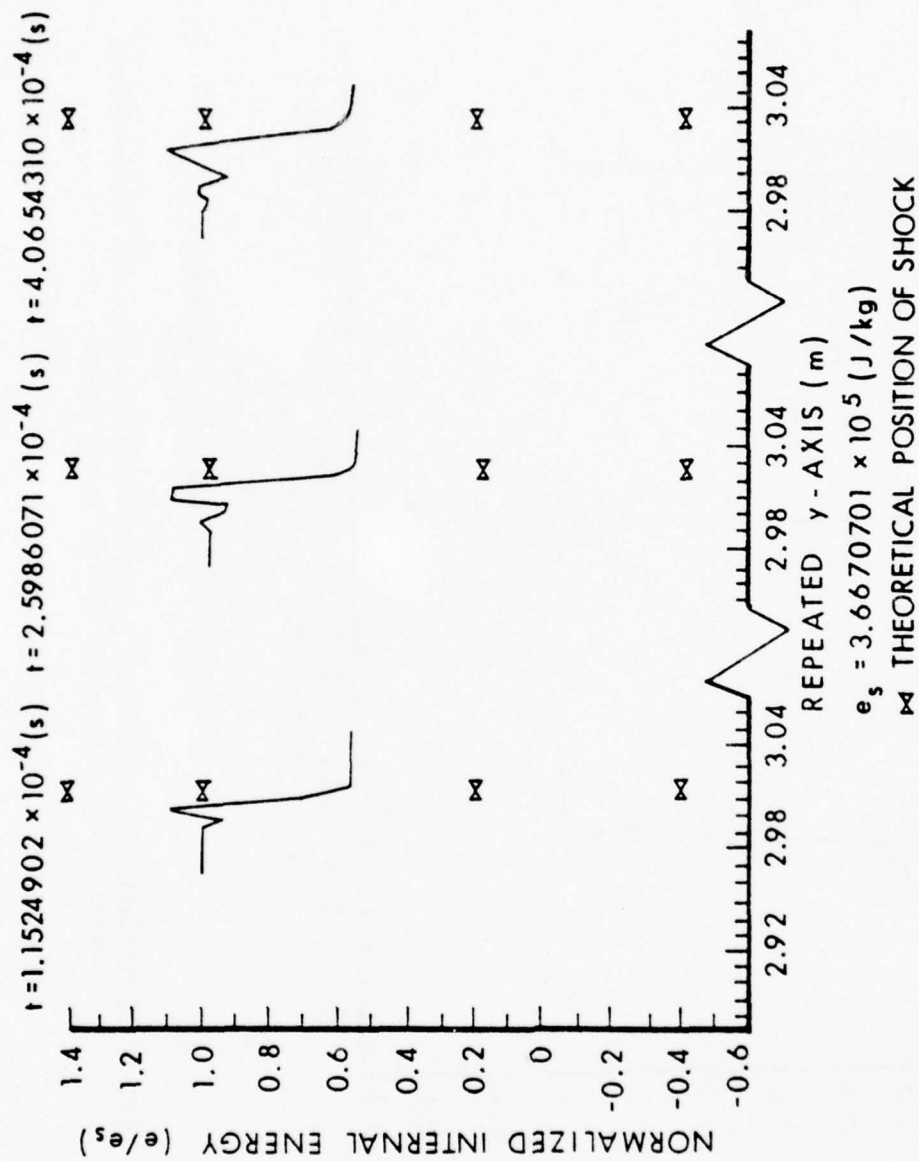


Figure 9. Variation of Normalized Internal Energy with Distance for Normal Shock of Strength Five at Three Times.

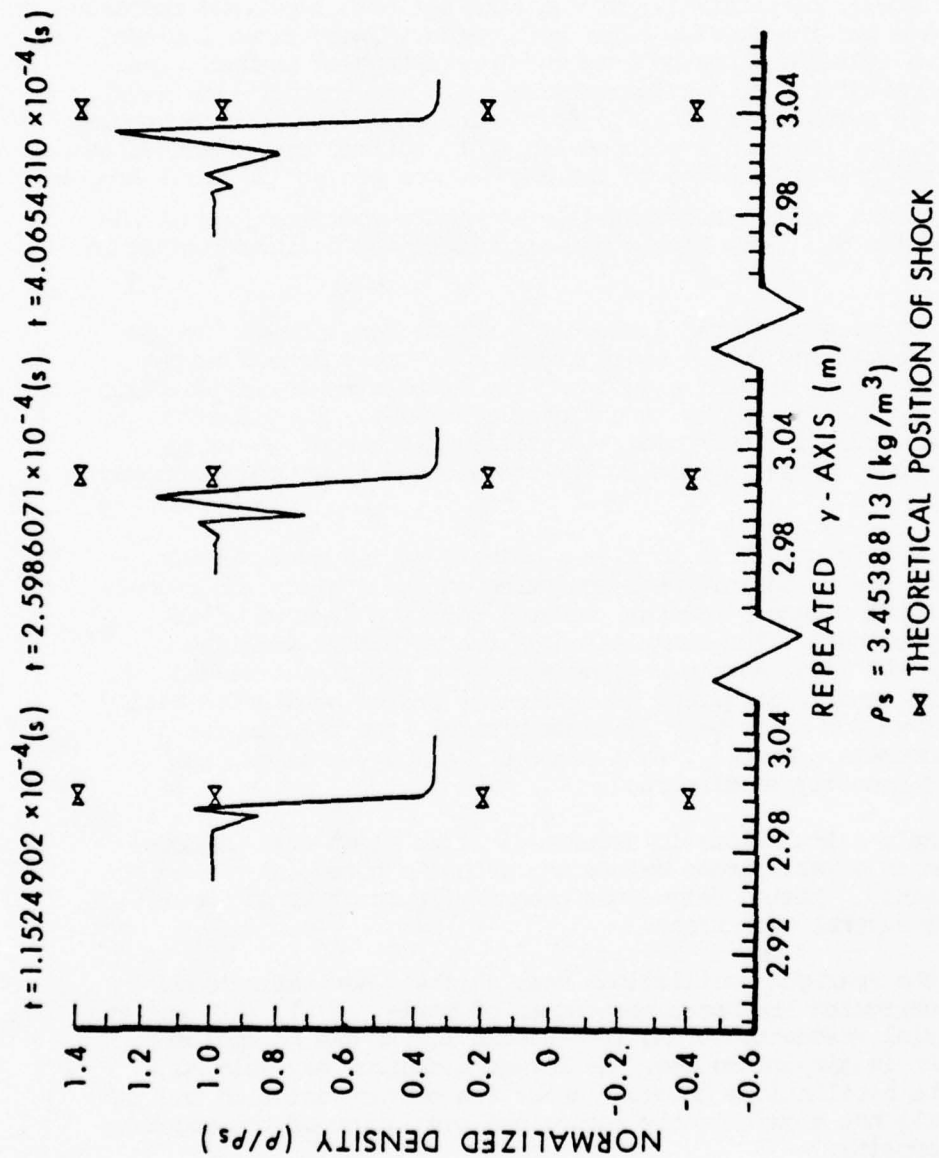


Figure 10. Variation of Normalized Density with Distance for Normal Shock of Strength Five at Three Times.

region. A major objective of future work is to curtail these oscillations (see section 5).

We close this section with a discussion of the calculation times. The times needed to calculate the flow variables both for an entire time level and at an individual interior node are given to help ascertain the time characteristics of the scheme. All the calculations were done on the BRLESC computer facility at the Ballistic Research Laboratory. The calculations of the flow behind a cylindrical blast wave took approximately 1.3 and 0.5 minutes for the finer and coarser grids, respectively, of run time per time level and approximately eight seconds per interior node for both grids. These times include, on the average, two iterations per node in the Newton-Raphson method. The shock propagation calculations took approximately 4.5 minutes per time level and approximately 16 seconds per interior node. The shock propagation problem typically required four iterations per node in order to find convergent values of the unknown parameters a_1 . Although the above times are not as small as one might hope, they can be reduced substantially by simple modifications of the algorithm (see section 5). Such simple optimizations must be investigated in future work.

5. SUMMARY. We have presented a numerical scheme for solving time dependent hyperbolic equations in two space dimensions. This method merges the concepts of the finite element method and the properties of hyperbolic systems of equations and is applied to unsteady gas flows. The essential features of the corresponding hydrocode are briefly discussed. Finally, numerical calculations involving both smooth and shocked flows are given and discussed.

The purpose of this report is to give a summary of the work already accomplished rather than a definitive description of the quality and usefulness of this numerical scheme. However, several positive aspects of the method can already be seen. The formulation of the method is straightforward and avoids the large matrices associated with the finite element method. The method handles different geometrically shaped boundaries easily and accurately. Even for relatively large mesh sizes, the results for a smooth flow are accurate. Finally, in a propagating shock problem, the shock's position can easily be discerned.

From the example calculations in section 4, it is clear that improvements must be made in several areas before the method's potential can be accurately ascertained. Listed below, not necessarily in order of importance or difficulty, are several such areas.

a. Curtail the spurious oscillations near shocks. Several methods; such as the flux-corrected transport techniques of Boris, et al, [11] and the well-known artificial viscosity method (see Roache [12]), can be applied. Although the latter is simpler to use, the former technique may hold more promise because the oscillations do occur about the correct solution and the flux correction will not significantly reduce the resolution of the shock as does artificial viscosity.

b. Shorter computing time. The run time can be significantly reduced by simple alterations in the algorithm. Recall that the spatial derivatives of the variables at the known time level (u_0, v_0, p_0, e_0) were computed on every triangular base of the finite element which resulted in the repeated calculation (up to NUMTRI times) of the residuals D_k and their partial derivatives at the nodes (see equations (13) - (14)). If these derivatives at the nodes were computed once for a given finite element, the time reduction would be by a factor of 0.5. The run time could be farther reduced by, at least, an order of magnitude if the calculations were done on a machine similar to the CDC 7600.

c. Rerun examples with different equation formulations. Certain formulations may increase the accuracy of the computed results and decrease the oscillations due to shocks.

d. Apply the method to an actual numerical problem. By applying the method to a problem with complicated boundaries, not only could the method's treatment of boundaries be tested, but also the entire method could be compared to another numerical solution technique.

e. Extend the method with respect to "infinite" strength shocks and moving boundaries. Once these extensions are incorporated into the method, this scheme could hopefully be used to finally develop an adequate model of the severe transitional ballistics environment.

ACKNOWLEDGEMENT

The author wishes to thank Dr. Aivars Celmiņš for his continual help and guidance during the course of this investigation.

REFERENCES

1. O. C. Zienkiewicz, The Finite Element Method in Engineering Science, McGraw Hill, 1971.
2. J. T. Oden, O. C. Zienkiewicz, R. H. Gallagher, C. Taylor, eds., Finite Element Methods in Flow Problems, University of Alabama in Huntsville Press, 1974.
3. P. P. Lynn and S. K. Arya, "Use of the Least Squares Criterion in the Finite Element Formulation", *Int. J. Num. Meth. Engng.*, 6, 75-88, 1973.
4. P. P. Lynn and S. K. Arya, "Finite Elements Formulated by the Weighted Discrete Least Squares Method", *Int. J. Num. Meth. Engng.*, 8, 71-90, 1974.
5. J. Polk, "A Least Squares Finite Element Approach to Unsteady Gas Dynamics", BRL Report No. 1885, May 1976.
6. J. T. Oden, L. C. Wellford, and C. T. Reddy, "A Study of Convergence and Stability of Finite Element Approximation of Shock and Acceleration Waves in Nonlinear Materials", U. S. Army Research Office Report P-11860-M/DAAG29-76-G-0022, August 1976.
7. J. A. Schmitt, "A Finite Element Method and Corresponding Pilot Computer Code for Hyperbolic Systems of Equations in Two Spatial Dimensions and Time Applied to Unsteady Gas Flows", BRL Report in preparation.
8. Gino Moretti, "A Pragmatical Analysis of Discretization Procedures for Initial- and Boundary-Value Problems in Gas Dynamics and Their Influence on Accuracy or Look Ma, No Wiggles!", Polytechnic Institute of New York, Report No. 74-15, September 1974.
9. L. A. Sedov, Similarity and Dimensional Methods in Mechanics, Academic Press, 1959.
10. H. W. Liepmann and A. Roshko, Elements of Gasdynamics, Wiley and Sons, 1957.
11. D. L. Book, J. P. Boris, and K. Hain, "Flux-Corrected Transport II: Generalizations of the Method", *J. Comp. Phys.*, 18, pp. 248-83, July 1975.
12. P. J. Roache, Computational Fluid Dynamics, Hermosa Publishers, P. O. Box 8172, Albuquerque, New Mexico, 1976.

APPENDIX A

BOUNDARY NODE FORMULATION FOR A ZERO NORMAL VELOCITY BOUNDARY CONDITION

Let Q be a node at a stationary wall where the tangent is defined. The normal velocity being zero at Q implies that

$$u(\bar{x}, \bar{y}, t) \sin \alpha - v(\bar{x}, \bar{y}, t) \cos \alpha = 0, \quad (A1)$$

where (\bar{x}, \bar{y}) are the spatial coordinates of node Q and α is the angle between the tangent line at Q and the x-axis. Since the wall is stationary α is a constant and equation (A1) holds for all times. By translating the axis to $(\bar{x}, \bar{y}, \bar{t})$, (the plane corresponding to \bar{t} is the initial plane), and using the interpolating functions for u and v, equations (9) and (10), respectively, equation (A1) becomes

$$\left[u_0(0,0,0) + a_3 t \right] \sin \alpha - \left[v_0(0,0,0) + a_6 t \right] \cos \alpha = 0, \quad 0 \leq t \leq \Delta t. \quad (A2)$$

Since the solid wall is stationary, equation (A2) reduces to

$$a_3 \sin \alpha - a_6 \cos \alpha = 0. \quad (A3)$$

To minimize the least square residual error over the finite element at the point $(\bar{x}, \bar{y}, \bar{t}, \Delta t)$ subject to equation (A3), we use Lagrange multipliers to obtain

$$\begin{aligned} \cos \alpha \left[\sum_{\ell=1}^{\text{NUMTRI}} \int_{\ell} \int_{\text{prism}} \sum_{k=1}^4 D_k \frac{\partial D_k}{\partial a_3} dx dy dt \right] \\ + \sin \alpha \left[\sum_{\ell=1}^{\text{NUMTRI}} \int_{\ell} \int_{\text{prism}} \sum_{k=1}^4 D_k \frac{\partial D_k}{\partial a_6} dx dy dt \right] = 0, \end{aligned} \quad (A4)$$

$$a_3 \sin \alpha - a_6 \cos \alpha = 0. \quad (A5)$$

In the minimization of the residual error over the boundary type element at which zero radial velocity is imposed, equations (A4) and (A5) replace equations $\partial E(\vec{a})/\partial a_3 = 0$ and $\partial E(\vec{a})/\partial a_6 = 0$ of equation system (13), respectively.

LIST OF SYMBOLS

\vec{a}	vector of unknown parameters
a_i	i^{th} component of vector \vec{a}
$e(x, y, t)$	internal energy per unit mass [J/kg]
$p(x, y, t)$	pressure [Pa]
r	polar radial coordinate [m]
r_a, r_b	radii of annular region for blast wave calculation [m]
t	time [s]
\bar{t}	given value of time [s]
Δt	time increment [s]
t_m	Gaussian quadrature points used on time integration [s]
$u(x, y, t)$	velocity component in x direction [m/s]
$v(x, y, t)$	velocity component in y direction [m/s]
$v_r(x, y, t)$	velocity component in the radial direction [m/s]
x	Cartesian spatial coordinate [m]
\bar{x}	given value of the x coordinate [m]
y	Cartesian spatial coordinate [m]
\bar{y}	given value of the y coordinate [m]

$D_k(x, y, t; \vec{a})$	nondimensional residual error from the k^{th} differential equation at a point (x, y, t)
$E(\vec{a})$	total residual least squares error over a finite element
$F_i(\vec{a})$	the first partial derivative of $E(\vec{a})$ with respect to a_i
NOEQ	number of differential equations to be solved simultaneously
NUMTRI	number of prisms composing a finite element
V	volume of a finite element [m^3]
$\rho(x, y, t)$	density [kg/m^3]
$\omega(x, y, t)$	generic flow variable
Subscripts	
o	corresponds to known value at a given time
s	corresponds to value at shock front
1	corresponds to value in the pre-shock state
2	corresponds to value in the post-shock state

THE NUMERICAL SOLUTION OF AXISYMMETRIC FREE
BOUNDARY POROUS FLOW WELL PROBLEMS

Colin W. Cryer and Hans Fetter

Mathematics Research Center, University of Wisconsin, Madison, Wisconsin

1. Introduction

The steady state problem to be considered is shown in Figure 1.1. An axisymmetric well of radius r is sunk into a layer of soil of depth H and radius R . The bottom of the soil layer is impervious. The outer boundary of the soil adjoins a catchment area and the hydraulic head u is equal to the constant H along this boundary. The water seeps towards the well and a pump (not shown) maintains the water level in the well at a constant height h_w . The water-air interface is a free boundary which intersects the well wall at a height h_s .

The mathematical problem can now be formulated as follows (see Hantush [1964], Bear [1972], and Cryer [1976, p. 86]):

Problem 1.

Find functions $\varphi(x)$ (the height of the free boundary) and $u(x,y)$ (the hydraulic head) such that (from the equation of continuity and Darcy's law):

$$\operatorname{div}(k \operatorname{grad} u) = \frac{\partial}{\partial x} \left(k \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial u}{\partial y} \right) = 0, \quad \text{in } \Omega, \quad (1.1)$$

together with the boundary conditions,

$$u = H, \quad \text{on } AB(\Gamma_1), \quad (\text{constant hydraulic head}), \quad (1.2)$$

$$\frac{\partial u}{\partial n} = 0, \quad \text{on } BC(\Gamma_4), \quad (\text{impervious boundary}), \quad (1.3)$$

$$u = h_w, \quad \text{on } CD(\Gamma_2), \quad (\text{interface with water at rest}), \quad (1.4)$$

$$u = y, \quad \text{on } DE(\Gamma_3'), \quad (\text{interface with air}), \quad (1.5)$$

$$u = y, \quad \text{on } EA(\Gamma_0), \quad (\text{interface with air}), \quad (1.6)$$

$$\frac{\partial u}{\partial n} = 0, \quad \text{on } EA(\Gamma_0), \quad (\text{streamline}). \quad (1.7)$$

Sponsored by

- 1) the United States Army under Contract No. DAAG29-75-C-0024,
- 2) the National Science Foundation under Grant No. DCR75-03838.

Here, Ω is the (unknown) domain,

$$\Omega = \{(x,y): 0 < y < \varphi(x), \quad x < x < R\},$$

and $\frac{\partial}{\partial n}$ denotes the outward normal derivative. Finally, $k = k(x,y) = \lambda$ where the permeability of the soil is denoted by $\kappa = \kappa(x,y)$. It is assumed, that k is of the form

$$k(x,y) = \exp[f(x) + g(y)] \quad (1.8)$$

where $f(x)$ and $g(y)$ are continuously differentiable and

$$g'(y) \geq 0. \quad n \quad (1.9)$$

In particular if the permeability κ is constant, $\kappa = 1$ say, then

$$k(x,y) = \lambda = \exp[\ln x]$$

so that

$$f(x) = \ln x; \quad g(y) = 0. \quad (1.10)$$

Ever since C. Baiocchi [1971] introduced a mathematically rigorous and from the numerical point of view efficient approach to the solution of various free boundary problems related to fluid flow through porous media, numerous studies have appeared in the literature which extend his results in many directions; Cryer [1976] and Baiocchi, Brezzi, and Comincioli [1976] give bibliographies.

The basic idea introduced by Baiocchi, can be summarized as follows: Through a suitable change of the unknown variable, the free boundary problem is reduced to that of minimizing a quadratic functional on a closed convex set. This reformulation of the problem not only enables one to determine various properties of the solution, but it also offers the advantage that the new problem can readily be solved numerically by several methods including finite differences and finite elements.

Here, we apply an extension of Baiocchi's work due to Benci [1973,1974]. A more complete account of our work will be found in Cryer and Fetter [1977].

2. Formulation as a variational inequality

In this section we follow Benci [1973, 1974] and reformulate Problem A, as a variational inequality.

The first step is to introduce a "Baiocchi function" $w(x,y)$ defined on the rectangle

$$D = \{(x,y): r \leq x \leq R, 0 \leq y \leq H\}, \quad (2.1)$$

as follows:

$$w(x,y) = \begin{cases} \int_y^{\varphi(x)} \exp(g(t)) [u(x,t) - t] dt, & \text{for } (x,y) \in \Omega, \\ 0, & \text{for } (x,y) \in D - \Omega. \end{cases} \quad (2.2)$$

The values of w on the boundary of D can be given explicitly; for a special case see (3.8).

We use the following function spaces (see Adams [1975]): $C(\bar{D})$, the space of functions which are continuous on \bar{D} , equipped with the supremum norm; $H^1(D)$, the Sobolev space of weakly differentiable functions on D , which is sometimes denoted by $H^{1,2}(D)$ or $W^{1,2}(D)$, and which is equipped with the norm $\|\cdot\|_{1,2}$; and $H_0^1(D)$, the subspace of $H^1(D)$ consisting of those elements of $H^1(D)$ which "vanish" on the boundary of D .

Let a be the bilinear operator defined on $H^1(D) \times H^1(D)$ by,

$$a(u,v) = \int_D \exp[f(x) - g(y)] \text{grad } u \text{ grad } v \, dx dy, \quad (2.3)$$

$$\equiv \int_D \exp[f(x) - g(y)] \left[u_x v_x + u_y v_y \right] dx dy$$

Let j be the linear functional defined on $H^1(D)$ by,

$$j(v) = \int_D \exp[f(x)] v \, dx dy \quad (2.4)$$

Let K be the closed convex set

$$K = \{v \in H^1(D) : v - w \in H_0^1(D) \text{ and } v \geq 0 \text{ a.e. in } D\} . \quad (2.5)$$

Then Benci [1973, 1974] proves

Theorem 2.1

If u is a solution of Problem A and $g'(y) \geq 0$ then $w \in H^1(D) \cap C(\bar{D})$ and w satisfies the variational inequality: Find $w \in K$ such that

$$a(w, v-w) + j(v-w) \geq 0 , \quad (2.6)$$

for all $v \in K$. \square

It follows from the basic theory of variational inequalities (Stampacchia [1964]) that

Theorem 2.2

There exists a unique solution $w \in H_0^1(D)$ of the variational inequality formulation (2.6) of the axisymmetric well problem. \square

3. Numerical approximation

It has been shown in the previous section that the Baiocchi function w satisfies the variational inequality (2.6): Find $w \in K$ such that for all $v \in K$,

$$a(w, v-w) + j(v-w) \geq 0 . \quad (3.1)$$

Since a is symmetric, that is

$$a(v, w) = a(w, v), \text{ for all } v, w \in V ,$$

there is a connection between the variational inequality (3.1) and the unilateral minimization problem

$$\begin{aligned} & \text{Min } J(v) , \\ & v \in K \end{aligned} \quad (3.2)$$

$$J(v) = a(v, v) + 2j(v) .$$

This connection is given by the following theorem (Lions [1971, p. 9]):

Theorem 3.1

Let $a(v, w)$ be a symmetric coercive bilinear form. Then w is a solution of the variational inequality (3.1) iff w is a solution of the unilateral minimization problem (3.2).

\square

We approximate w by choosing a finite-dimensional approximation K_h and solving the finite-dimensional problem: Find $w_h \in K_h$

$$J(w_h) = \min_{v_h \in K_h} J(v_h) . \quad (3.3)$$

The convex set K_h is constructed as follows. The domain D is triangulated as shown in Figure 3.1.

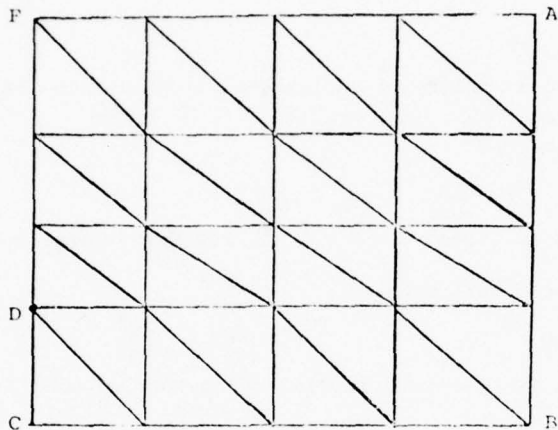


Figure 3.1: The triangulation of D

The subdivisions are not necessarily uniform, but it is assumed that there is a constant $\beta > 0$ such that

$$\frac{1}{\beta} (\text{maximum interval length}) \leq h \leq \beta (\text{minimum interval length}), \quad (3.4)$$

where h is a measure of the fineness of the subdivision. The set of interior gridpoints will be denoted by D_h and the set of boundary gridpoints will be denoted by ∂D_h .

We denote by V_h the space of piecewise linear functions (linear finite elements) v_h corresponding to the triangulation in Figure 3.1. We set

$$K_h = \{v_h \in V_h : v_h \geq 0 \text{ in } D \text{ and } v_h = \phi \text{ on } \partial D_h\} . \quad (3.5)$$

The approximation w_h is readily computed. We can derive an error estimate for $\|w - w_h\|$ by combining the ideas of Brezzi and Sacchi [to appear] and Brezzi, Hager, and Raviart [to appear]:

Theorem 3.2

The piecewise linear approximate solution w_h exists and is unique. Furthermore,

$$\|w - w_h\|_{1,2} = O(h) \quad \square \quad (3.6)$$

As an example we consider the specific geometry shown in Figure 3.2, which was chosen because it had previously been considered by several authors.

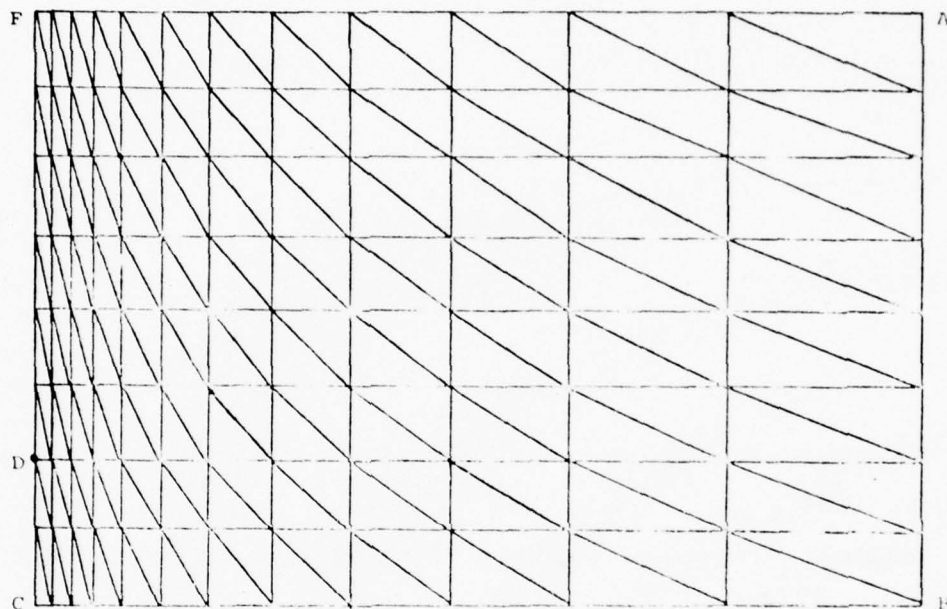


Figure 3.2: A numerical example ($r = 4.8$, $R = 76.8$, $h_w = 12$, $H = 48$, $m = 8$, $n = 12$.)

Because the solution changes most rapidly near the well, the subdivisions were taken to be uniform in the y -direction and logarithmic in the x -direction. If n and m denote the number of subdivisions in the x - and y -directions, the coordinates of the gridpoints were given by

$$y_j = j H/m, \quad 0 \leq j \leq m,$$

$$x_i = r \exp[i/n \ln(R/r)], \quad 0 \leq i \leq n$$

The integer m was always chosen to be a multiple of 4 so that the corner D was a gridpoint; this was advisable since w is not smooth at the corner D .

The permeability K was taken to be one so that $f(x) = \ln x$ and $g(y) = 0$. From (2.3), (2.4), and (3.2),

$$J(v) = \int_D x[v_x^2 + v_y^2 + 2v] dx dy. \quad (3.7)$$

The boundary values of w are

$$\begin{aligned} w(x, H) &= 0, \quad \text{on } AF, \\ w(R, y) &= (H-y)^2/2, \quad \text{on } AB, \\ w(r, y) &= (h_w - y)^2/2, \quad \text{on } CD, \\ w(r, y) &= 0, \quad \text{on } DF, \\ w(x, 0) &= \frac{h_w^2 \ln(R/x) + H^2 \ln(x/r)}{2 \ln(R/r)}, \quad \text{on } BC. \end{aligned} \quad (3.8)$$

The approximation w_h was computed using S.O.R. with projection (Cryer [1971], Glowinski [1971]).

The computations presented no difficulties. The solution of the smallest problem is given in Table 3.1.

In Table 3.1 the position of the approximate free boundary is shown by the first zero term in each column. The approximate solution is identically zero on the vertical line $x = r$ so that it is not possible to determine the height h_s at which the free boundary intersects the well. As an approximation to h_s we take the height h'_s of the free boundary at the vertical gridline adjacent to the well. For example, from Table 3.1 we obtain $h'_s = 36$.

For comparison, we compare in Table 3.2 the values of h_s obtained by different authors. With the exception of the present computation, all the results are presented graphically so that we have had to estimate h_s from graphs.

x

y	4.8000000,+00	7.6195250,+00	1.2095242,+01	1.9200000,+01	3.0478100,+01	4.8380968,+01	7.6800000,+01
48.00	0	0	0	0	0	0	0
36.00	0	0	0	0	3.0332990,+00	3.1593876,+01	7.2000000,+01
24.00	0	1.7554644,+01	4.3617047,+01	8.1333841,+01	1.3309081,+02	2.0406162,+02	2.8800000,+02
12.00	0	8.8585977,+01	1.8202316,+02	2.8314043,+02	3.9437102,+02	5.1710713,+02	6.4800000,+02
0.0000	7.2000000,+01	2.5200000,+02	4.3200000,+02	6.1200000,+02	7.9200000,+02	9.7200000,+02	1.1520000,+03*

Table 3.1: Solution for $m = 4, n = 6$

Author	Method	h_s
Hall [1955, p. 29]	trial-free-boundary; finite differences	34.0
Taylor and Luthin [1969]	time-dependent; finite differences	34.0
Neuman and Witherspoon [1970]	trial free-boundary; finite elements	30.0
Neuman and Witherspoon [1971, p. 620]	time-dependent; finite differences	30.0
Present (m=64, n=36)	Variational inequalities	30.0

Table 3.2: Computed values of h_s

The differences in Table 3.2 may be explained by the fact that the physical assumptions differed: Hall [1955] assumed capillarity and a lined well; Taylor and Luthin [1969] assumed partially saturated flow; and Neuman and Witherspoon [1970, 1971] made the same assumptions as in the present paper.

References

- ADAMS, R. A.: Sobolev Spaces. New York: Academic Press, 1975.
- BAIOCCHI, C.: Sur un problème à frontière libre traduisant le filtrage de liquides à travers des milieux poreux. Comptes. Rendus Acad. Sci. Paris A273, 1215-1217 (1971).
- BAIOCCHI, C., BREZZI, F. and COMINCIOLI, V.: Free boundary problems in fluid flow through porous media. In Proceedings Second International Symposium on Finite Element Methods in Flow Problems, Santa Margherita, Italy, 1976.
- BEAR, J.: Dynamics of Fluids in Porous Media. New York: American Elsevier, 1972.
- BENCI, V.: Su un problema di filtrazione in un mezzo poroso non omogeneo. Rend. Acad. Naz. Lincei (8) 54, 10-15 (1973).
- BENCI, V.: On a filtration problem through a porous medium. Annali di Matem. (4) 100, 191-209 (1974).
- BREZZI, F., HAGER, W. H., and RAVIART, P. A.: Error estimates for the finite element solution of variational inequalities Part I: Primal theory. To appear.
- BREZZI, F. and SACCHI, G.: A Finite element approximation of a variational inequality related to hydraulics. Pubblicazioni No. 97, Laboratorio di Analisi Numerica del C.N.R., Università di Pavia, to appear.
- CRYER, C. W.: The solution of a quadratic programming problem using systematic over-relaxation. SIAM J. Control 9, 385-392 (1971).
- CRYER, C. W.: A survey of steady-state porous flow free boundary problems. Technical Summary Report No. 1657, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin, 1976.
- CRYER, C. W. and FETTER, H.: The numerical solution of axisymmetric free boundary porous flow well problems using variational inequalities. Technical Summary Report, Mathematics Research Center, University of Wisconsin, 1977.
- GLOWINSKI, R.: La méthode de relaxation. Rendiconti di Matematica 14, Università Roma, (1971).
- HALL, H. P.: An investigation of steady flow towards a gravity well. La Houille Blanche 10, 8-35 (1955).

- HANTUSH, M. S.: Hydraulics of wells. *Advances of Hydroscience* 1, 281-432 (1964).
- LIONS, J.L.: Optimal Control of Systems Governed by Partial Differential Equations.
Berlin: Springer, 1971
- NEUMAN, S. P. and WITHERSPOON, P. A.: Finite element method of analyzing steady seepage with a free surface. *Water Resources Res.* 6, 889-897 (1970).
- NEUMAN, S. P. and WITHERSPOON, P. A.: Analysis of nonsteady flow with a free surface using the finite element method. *Water Resources Res.* 7, 611-623 (1971).
- STAMPACCHIA, G.: Formes bilinéaires coercitives sur les ensembles convexes. *Comptes. Rendus Acad. Sci. Paris* 258, 4413-4416 (1964).
- TAYLOR, G. S. and LUTHIN, J. N.: Computer methods for transient analysis of water-table aquifers. *Water Resources Res.* 5, 144-152 (1969).

DIRECT AND ITERATIVE ONE DIMENSIONAL FRONT TRACKING METHODS
FOR THE TWO DIMENSIONAL STEFAN PROBLEM

Gunter H. Meyer
School of Mathematics
Georgia Institute of Technology
Atlanta, Georgia 30332

ABSTRACT. An application of the method of fractional steps and of the method of lines to general two dimensional Stefan type problems is described. The numerical aspects are discussed and the performance of both methods for a model problem is compared.

1. INTRODUCTION. Two and three dimensional free surface problems for partial differential equations lead to many of the same computational difficulties observed with nonlinear boundary value problems on fixed domains. The data management demands and long computing times required for the iterative solution of nonlinear problems place a premium on the efficiency of the numerical method itself and on its implementation on a given computer installation. If a general class of problems can be identified which must be solved repeatedly, then a careful design of a computer algorithm and its code (and the concomitant expense) is justified. If, however, nonstandard problems are to be solved then an ad-hoc method with modest programming demands may be attractive even if computing times are not minimized. In our view free surface problems may be considered nonstandard because of the variety of differential operators and boundary conditions which arise in practice; and locally one dimensional methods are advocated as solution techniques precisely because they place modest demand on mathematical training and programming skill. They may not be competitive in execution time with more sophisticated techniques (such as multi dimensional finite element methods), but such differences blur when weighted against the accounting practice for man and machine of any given computer center.

Two distinct locally one dimensional methods have been examined so far [3], [4], [5]. A naive version of a fractional step method (a special alternating direction method) is particularly simple to apply. It is effective in certain geometries and reasonably efficient computationally. An alternate approach is via the method of lines coupled with an iterative solution procedure. It is advocated for some geometries where the fractional step method cannot immediately be applied. While this method is also straightforward to explain it appears computationally less efficient for problems requiring a fine spatial resolution. In this paper both methods are presented for a general class of free boundary problems in the plane. More general differential operators and different boundary conditions are considered here than in the earlier papers. Moreover, a first comparison is made between both methods in the solution of a realistic problem when each one dimensional problem is solved with the front tracking invariant imbedding method.

Research supported by the U. S. Army Research Office under Grant DA-AG29-76-G-0261

2. THE METHOD OF FRACTIONAL STEPS FOR FREE SURFACE PROBLEMS. Primarily in the Russian literature the method of fractional steps has found favor as a straightforward solution technique for multi dimensional boundary value problems. An exposition of this method and selected applications may be found in the monograph of Yanenko [8]. Moreover, it is known that the method is valuable for certain nonlinear problems for parabolic equations, among them the classical Stefan problem in its enthalpy formulation which eliminates the need to track a priori unknown free surfaces [1], [7]. However, in many applications the differential equations and the boundary conditions make it very doubtful that fixed domain transformations similar to the enthalpy transformation exist at all so that it may become necessary to actually track the free surface as a function of time. It now seems natural to combine one dimensional front tracking routines with the method of fractional steps to resolve such problems via a sequence of (usually well understood) one dimensional problems. It has been demonstrated in [3] that an effective and transparent solution method for quite general free boundary problems results. We shall give here an exposition which somewhat parallels that of [3]; however, more general equations and flux boundary conditions rather than Dirichlet data are chosen to illustrate the versatility of the method.

The problem to be considered will be written as

$$(2.1) \quad Lu \equiv \nabla \cdot k \nabla u + \bar{a} \cdot \nabla u + bu - cu_t = f, \quad (x, y) \in D(t), \quad t > 0$$

$$(2.2) \quad u(x, y, 0) = u_0(x, y), \quad (x, y) \in D(0)$$

where the data functions k , $\bar{a} = (a_1, a_2)$, b , c , and f are assumed to be smooth functions of x , y , and t . The initial domain $D(0)$ is given; it evolves into the general domain $D(t)$ whose boundary in part is unknown. Specifically, we shall assume that the boundary $\partial D(t)$ of $D(t)$ consists of two parts $\partial D_1(t)$ and $\partial D_2(t)$ where $\partial D_1(t)$ is a given curve including its end points, and where $\partial D_2(t)$ is the free boundary. Quite general domains and boundary shapes for $\partial D_1(t)$ can be treated. However, to simplify the exposition and to aid in visualizing the geometry we shall assume that $D(t)$ consists of a rectangle $[0, X] \times [0, Y]$ whose upper right hand corner has been cut off by the free surface $\partial D_2(t)$. The numerical example of section 4 uses this geometry (see Fig. 1). The following assumption is essential for the applicability of the fractional step method:

Hypothesis: The free surface has no horizontal or vertical tangents.

This hypothesis implies that $\partial D_2(t)$ at any time t can be expressed in parametric form as

$$\partial D_2(t) = \{(x, s_2(x, t))\}$$

or

$$\partial D_2(t) = \{(s_1(y, t), y)\}$$

where $x = s_1(y, t)$ and $y = s_2(x, t)$ are inverse functions of each other. For Dirichlet data the end points of $\partial D_1(t)$ are given, while for more general flux conditions they must be calculated. The initial shape $\partial D_2(0)$ is assumed given.

On $\partial D_1(t)$ a reflection condition of the form

$$(2.3) \quad -k \frac{\partial u}{\partial n} = \eta u + \alpha \quad (x, y) \in \partial D_1(t)$$

will be imposed where again k , η and α are functions of x, y , and t . Since it is intended to apply invariant imbedding for each locally one dimensional problem the specification of the data for the free boundary $\partial D_2(t)$ can be kept quite general. Indeed, as indicated in [3], the boundary conditions on $\partial D_2(t)$ are important only in so far as they lead to an orderly evolution of $\partial D_2(t)$ obeying the above hypothesis. Accordingly, we shall write

$$(2.4) \quad u = g_0(x, y, t) \quad (x, y) \in \partial D_2(t)$$

$$(2.5) \quad k \nabla u = (g_1(x, y, \frac{dx}{dt}, t), g_2(x, y, \frac{dy}{dt}, t))$$

(A specific example of a heat flow problem with space dependent energy input on the free surface is presented below to illustrate the meaning of these conditions.)

In the method of fractional steps the equation (2.1) is written as

$$(2.6) \quad Lu \equiv L_1 u + L_2 u = f$$

where $L_1 u = (ku_x)_x + a_1 u_x + \frac{1}{2} b u - \frac{1}{2} c u_t$

and $L_2 u = (ku_y)_y + a_2 u_y + \frac{1}{2} b u - \frac{1}{2} c u_t$,

and its solution is advanced over a discrete time step of length Δt by integrating each one dimensional equation $L_1 u = \frac{1}{2} f$ and $L_2 u = \frac{1}{2} f$ over successive half steps. Specifically, suppose that $u_n \equiv u(x, y, t_n)$ is known over the given domain $D(t_n)$. Then the solution u and domain $D(t_{n+\frac{1}{2}})$ at the time level $t = t_n + \frac{1}{2} \Delta t \equiv t_{n+\frac{1}{2}}$ are found from the one dimensional problem

$$(2.7) \quad L_1 u = \frac{1}{2} f \quad (x, y) \in D(t), \quad t \in (t_n, t_{n+\frac{1}{2}}]$$

$$k(x, y, t) \frac{\partial u}{\partial x} = \eta(x, y, t) u + \alpha(x, y, t), \quad (x, y) \in \partial D_1(t)$$

$$u = g_0(x, y, t) \quad (x, y) \in \partial D_2(t)$$

$$ku_x = g_1(x, y, \frac{dx}{dt}, t)$$

with $u_n \equiv u(x, y, t_n)$ as given on the known domain $D(t_n)$. In order to solve (2.7) the free boundaryⁿ is given as $x = s_2(y, t)$. Throughout these equations the variable y is considered a parameter. (Numerically, of course, this one dimensional problem must be solved for representative values of y .)

Once $u(x, y, t_{n+\frac{1}{2}})$ and $D(t_{n+\frac{1}{2}})$ have been found they serve as initial values for the second step in which the evolution of u and $D(t)$ over $(t_{n+\frac{1}{2}}, t_{n+1}]$ is

completed by solving

$$\begin{aligned}
 (2.8) \quad L_2 u &= \frac{1}{2} f & (x, y) \in D(t), \quad t \in (t_{n+\frac{1}{2}}, t_{n+1}] \\
 k(x, y, t) \frac{\partial u}{\partial y} &= \eta(x, y, t) u + \alpha(x, y, t) & (x, y) \in \partial D_1(t) \\
 u &= g_0(x, y, t) & (x, y) \in \partial D_2(t) \\
 k u_y &= g_2(x, y, \frac{dx}{dt}, t),
 \end{aligned}$$

again with $u(x, y, t_{n+\frac{1}{2}})$ as computed on $D(t_{n+\frac{1}{2}})$.

Note that if in (2.7) and (2.8) the lines of constant y and x do not intersect the free boundary $\partial D_2(t)$ then the one dimensional operators $L_1 u$ and $L_2 u$ are subject to standard flux two point boundary conditions. Thus the above formulation includes standard diffusion problems on fixed domains. Alternately, one may visualize $\partial D_2(t)$ as given and the two conditions (2.4,5) replaced by a single nonlinear relation between u and ∇u . In this way nonlinear boundary data on selected portions of the boundary can be accommodated. We shall not pursue this point here.

An analysis of the analytical fractional step method (2.7,8) as $\Delta t \rightarrow 0$ for general free surface problems is not available at this time. However, the extensive numerical experiments with Stefan like problems reported in [3] indicate that even a fairly crude numerical integration of the one dimensional equations (2.7) and (2.8) for discrete values of the parameters y and x can adequately reproduce a-priori known solutions. Therefore, always under the hypothesis of invertibility of the free surface, the following algorithm can be suggested with confidence for the numerical solution of (2.7,8).

Let X and Y be upper bounds on the position of the free surface along the x and y axes so that $D(t)$ will always be contained in the rectangle $[0, X] \times [0, Y]$. On this rectangle we shall place mesh lines $x = x_i$, $i = 0, \dots, M$ and $y = y_j$, $j = 0, \dots, N$. We then shall solve (2.7) along those lines $y = y_j$ which can be expected to intersect $D(t)$, and (2.8) along the lines $x = x_i$ which also intersect $D(t)$. Let us consider in detail the numerical solution of (2.7).

It is assumed that at the initial time $t = t_n$ the solution u is known at the mesh points $\{(x_i, y_j)\}$ of the grid on $[0, X] \times [0, Y]$. It is also assumed that the free boundary $x = s_2(y, t)$ is known along each line $y = y_j$ belonging to $D(t_n)$. However, it need not coincide with a mesh point. Along any line $y = y_j$ the solution $u(x, y_j, t)$ of (2.7) is advanced over the fractional time step $\Delta t/2$ in one numerical time step of equal length. (Conceivably, if the diffusion takes place primarily in one direction, an unequal weighting of the fractional steps and a refined time step for the numerical integration in the dominant direction may be preferable. So far, no such asymmetry in the method has been used.) Invariant imbedding will be applied to the time implicit approximation of (2.7) at $t_{n+\frac{1}{2}}$. We shall write (2.7) in first order form as

$$(2.9) \quad (ku') = v$$

$$v' = -\frac{a_1}{k} v - \frac{1}{2}bu + \frac{1}{2}c \left[\frac{u-u_n}{\Delta t/2} \right] + \frac{1}{2}f$$

where it is understood that all functions depend on the independent variable x and the parameters y_j and $t_{n+\frac{1}{2}}$. The appropriate boundary conditions are

$$(2.10) \quad v(0) = \eta(0, y_j, t_{n+\frac{1}{2}})u(0) + \alpha(0, y_j, t_{n+\frac{1}{2}})$$

and, if the line $y = y_j$ intersects the free boundary $\partial D_2(t)$,

$$(2.11a) \quad u(s_1) = g_0(s_1, y_j, t_{n+\frac{1}{2}})$$

$$(2.11b) \quad v(s_1) = g_1(s_1, y_j, \frac{s_1 - s_1(y_j, t_n)}{\Delta t/2}, t_{n+\frac{1}{2}})$$

where $s_1 = s_1(y_j, t_{n+\frac{1}{2}})$ is the unknown location of the free boundary $\partial D_2(t_{n+\frac{1}{2}})$ on the line $y = y_j$. If this line does not intersect $\partial D_2(t_{n+\frac{1}{2}})$ then these two equations are replaced by

$$(2.11c) \quad -v(X) = \eta(X, y_j, t_{n+\frac{1}{2}})u(X) + \alpha(X, y_j, t_{n+\frac{1}{2}}).$$

It may be noted that the equations (2.9-11) have to be solved along each line $y = y_j$, $j = 1, \dots, N$ but that they are uncoupled.

According to the invariant imbedding formalism we write

$$(2.12) \quad v = R(x)u + w(x)$$

where R and w are found from the initial value problems

$$(2.13a) \quad R' = -\frac{1}{2}b + \frac{c}{\Delta t} - \frac{a_1}{k}R - \frac{1}{k}R^2, \quad R(0) = \eta(0, y_j, t_{n+\frac{1}{2}})$$

$$(2.13b) \quad w' = -\left[\frac{a_1}{k} + \frac{1}{k}R \right]w - \frac{c}{\Delta t}u_n + \frac{1}{2}f, \quad w(0) = \alpha(0, y_j, t_{n+\frac{1}{2}})$$

Since u_n typically is only known at the mesh points x_i , $i = 0, \dots, M$ along the line $y = y_j$, an integration of (2.13) may require an interpolation of u_n between the mesh points. If the trapezoidal rule is used the integration of (2.13a and b) can be carried out by formula without the need for interpolation. This route is followed in the numerical computation below. In order to place the free boundary s_1 we observe from (2.12) and (2.11a and b) that s_1 must be chosen so that the relation

$$g_1(s_1, y_j, \frac{s_1 - s_1(y_j, t_{n+\frac{1}{2}})}{\Delta t/2}, t_{n+\frac{1}{2}}) = R(s_1)g_0(s_1, y_j, t_{n+\frac{1}{2}}) + w(s_1)$$

holds. Hence, as (2.13) is integrated we monitor the function

$$(2.14) \quad \phi(x) \equiv R(x)g_0(x, y_j, t_{n+\frac{1}{2}}) + w(x) - g_1(x, y_j, \frac{x-s_1(y_j, t_n)}{\Delta t/2}, t_{n+\frac{1}{2}})$$

and determine the root of

$$(2.15) \quad \phi(x) = 0$$

by linear interpolation between successive x mesh points between which ϕ changes sign for the first time. The validity of this procedure for Stefan problems is proven in [2]. If no such root is found on $[0, X]$ then this line does not intersect the free boundary and we can set $s_1 = X$. Finally, the solution u at $t = t_{n+\frac{1}{2}}$ along the line $y = y_j$ is formed by integrating backward from s_1 to 0 the equation

$$(2.15) \quad u' = \frac{R(x)u + w(x)}{k(x, y_j, t_{n+\frac{1}{2}})}$$

subject to the initial condition

$$u(s_1) = g_0(s_1, y_j, t_{n+\frac{1}{2}}) \quad \text{if } s_1 < X$$

or

$$u(X) = -\frac{(w(X) + \alpha(X, y_j, t_{n+\frac{1}{2}}))}{R(X) + \eta(X, y_j, t_{n+\frac{1}{2}})} \quad \text{if } s_1 = X$$

where the last condition is obtained by substituting (2.12) into (2.11c). Again, the trapezoidal rule can be used without interpolation. In general, a fractional step will be used to go from s_1 to the first regular x mesh point to the left of s_1 .

In order to integrate (2.8) along discrete lines $x = x_i$ over the time interval $[t_{n+\frac{1}{2}}, t_{n+1}]$ we proceed as above. An implicit time approximation is used to convert (2.8) into a free boundary problem for a second order linear ordinary differential equation. It is written as an equivalent first order system which looks like (2.9) except that a_1 is replaced by a_2 and u is replaced by $u_{n+\frac{1}{2}}$. All functions are now evaluated at x_i and $t_{n+\frac{1}{2}}$ and depend on the independent variable y . The equations (2.10) and (2.11) are replaced by

$$v(0) = \eta(x_i, 0, t_{n+\frac{1}{2}})u(0) + \alpha(x_i, 0, t_{n+\frac{1}{2}})$$

$$u(s_2) = g_0(x_i, s_2, t_{n+\frac{1}{2}})$$

$$v(s_2) = g_2(x_i, s_2, \frac{s_2-s_2(x_i, t_{n+\frac{1}{2}})}{\Delta t/2}, t_{n+\frac{1}{2}}).$$

$$\text{or} \quad -v(Y) = \eta(x_i, Y, t_{n+1})u(Y) + \alpha(x_i, Y, t_{n+1})$$

whenever the line $x = x_i$ does not intersect $\partial D_2(t)$.

If $u_{n+1/2}$ is needed outside $D(t_{n+1/2})$ to integrate (2.13b) it is extended linearly beyond the free boundary. A slight difficulty now arises. At time $t = t_{n+1/2}$ the free boundary is given as $x = s_1(y_i, t_{n+1/2})$ along each y line whereas now the inverse function s_2 is required. In our numerical method $y = s_2(x_i, t_{n+1/2})$ is obtained again by linear interpolation between successive y mesh points between which the expression $u(x_i, y, t_{n+1/2}) - g_0(x_i, y, t_{n+1/2})$ changes sign. For simple expressions of g_0 such as $g_0 \equiv 0$ the validity of such placement of the free boundary is easy to demonstrate. After $\partial D_2(t_{n+1/2})$ is found the inverse function $x = s_1(y_i, t_{n+1/2})$ of $y = s_2(x_i, t_{n+1/2})$ is found by interpolating and the integration over the next time step can begin.

3. THE METHOD OF LINES FOR FREE SURFACE PROBLEMS. In many applications the free surface cannot conveniently be expressed as an invertible curve with respect to two mutually orthogonal axes. This is the case, for example, whenever the movement takes place primarily along one of the coordinate axes. On the basis of the numerical experiments reported in [4], [5] it can be concluded that the method of lines discretization coupled with an iterative solution of the resulting multipoint free surface problem will also lead to a transparent and flexible numerical method. For definiteness let us again consider problem (2.1-5) with the understanding that the motion of the free boundary takes place along the x -axis. The hypothesis of section 2 can now be replaced by a weaker requirement.

Hypothesis: The free boundary has no horizontal tangents.

Under this condition the boundary $\partial D_2(t)$ can be expressed as

$$x = s_1(y, t).$$

In the method of lines algorithm discretizing y and replacing all derivatives with respect to y by difference quotients is suggested. Again, working on the square $[0, X] \times [0, Y]$ let $0 = y_0 < y_1 < \dots < y_N = Y$ define a partition which for convenience is chosen equidistant. Along the lines of constant y the equations (2.1-5) then are replaced by a system of coupled one dimensional equations. Specifically, along the line $y = y_j$ we shall solve

$$(3.1) \quad L_j u \equiv (k u_x)_x + a_1 u_x + (b - \frac{(k_{j+1/2} + k_{j-1/2})}{\Delta y^2})u - c u_t \equiv F(x, y_j, u_{j-1}, u_{j+1}, t)$$

$$\text{where } F(x, y_j, u_{j-1}, u_{j+1}, t) = (f - \frac{k_{j+1/2} u_{j+1} + k_{j-1/2} u_{j-1}}{\Delta y^2} - a_2 \frac{u_{j+1} - u_{j-1}}{2\Delta y}),$$

$$\text{where } k_{j \pm 1/2} = (k(x, y_{j \pm 1}, t) + k(x, y_j, t))/2,$$

and where all other data functions are evaluated at (x, y_j, t) . The approximation of $a_2 u_y$ by a central difference quotient was chosen for convenience only. Maximum principle arguments advanced for one sided quotients in finite difference methods

may apply here as well, but computational experience still is lacking. The initial and boundary conditions for (3.1) are

$$(3.2) \quad u(x, y_j, 0) = u_0(x, y_0)$$

$$(3.3) \quad k(0, y_j, t) \frac{\partial u}{\partial x} = \eta(0, y_j, t)u + \alpha(0, y_j, t)$$

$$(3.4) \quad u(s_1, y_j, t) = g(s_1, y_j, t)$$

$$(3.5) \quad k(s_1, y_j, t) u_x = g_1(s_1, y_j, \frac{ds_1}{dt}, t)$$

if the line $y = y_j$ intersects $\partial D_2(t)$, otherwise the two conditions (3.4,5) are replaced by

$$-k(X, y_j, t) \frac{\partial u}{\partial x} = \eta(X, y_j, t)u(X) + \alpha(X, y_j, t).$$

We see that formally this system is identical with (2.7) except that now a coupling to the solution along neighboring lines exists through the source term F . We remark that fractional step methods based on splittings analogous to (3.1) exist where the coupling then is removed through time lagging [8]. Here we intend to solve the fully coupled system through an SOR type iteration. Specifically, we shall solve over discrete time steps of length Δt and starting with a given initial condition $u(x, y, t_n)$ on a known domain $D(t_n)$ the following sequence

$$(3.6) \quad \begin{aligned} L_j \tilde{u} &= F(x, y_j, u_{j-1}^m, u_{j+1}^{m-1}, t) & t \in (t_n, t_n + \Delta t] \\ & & j = 0, \dots, N \\ u_j^m &= u_j^{m-1} + \omega(\tilde{u} - u_j^{m-1}) & m = 1, 2, \dots \end{aligned}$$

where \tilde{u} satisfies the boundary conditions (3.2-5) and the initial condition

$$\tilde{u}(x, y_j, t_n) = u(x, y_j, t_n).$$

For the initial guess we choose

$$u^0(x, y_j, t) = u(x, y_j, t_n)$$

since this initial condition appears to be of little importance to the calculation. The final solution $u(x, y_j, t)$ for $t \in [t_n, t_n + \Delta t]$ is obtained as

$$u(x, y_j, t) = \lim_{m \rightarrow \infty} u^m(x, y_j, t).$$

Computational evidence suggests the existence of this limit for Stefan like free boundary problems. A mathematical justification so far has been given only for certain nonlinear fixed domain problems [6].

The numerical solution of (3.6) proceeds exactly like that of (2.7). The solution $\{u, s_1\}$ along the line $y = y_j$ is advanced over one time step with the equations (2.12; 2.13a,b; 2.15 and 2.16) except that the following replacements are made:

$$t_{n+\frac{1}{2}} \leftarrow t_{n+1}; \quad \frac{1}{2}b \leftarrow (b - \frac{k_{j+\frac{1}{2}} + k_{j-\frac{1}{2}}}{\Delta y^2}); \quad \frac{1}{2}f \leftarrow F(x, y_j, u_{j-1}^m, u_{j+1}^{m-1}, t_{n+1})$$

and

$$\frac{x-s_1(y_j, t_n)}{\Delta t/2} + \frac{x-s_1(y_j, t_n)}{\Delta t}.$$

4. A NUMERICAL EXAMPLE. The existing computer codes of [3] and [4] for diffusion problems with Dirichlet data (rather than flux data) on the fixed boundary were modified to solve numerically the following two dimensional problem

$$\begin{aligned} (4.1) \quad & \Delta u - u_t = 0 & (x, y) \in D(t), \quad t > 0 \\ (4.2) \quad & u = u_0(x, y) & (x, y) \in D(0) \\ (4.3) \quad & u = \alpha(x, y, t) & \partial D_1(t) \quad [\{x=0\} \quad \{y=0\}] \\ (4.3') \quad & \frac{\partial u}{\partial n} = 0 & \partial D_1(t) \quad [\{x=1\} \quad \{y=1\}] \\ (4.4) \quad & u = 0 & (x, y) \in \partial D_2(t) \\ (4.5) \quad & \nabla u = - (g_1(x, y) + \frac{dx}{dt}, \quad g_2(x, y) + \frac{dy}{dt}) \end{aligned}$$

where $D(0) = [0, 1] \times [0, 1] - \{(x, y): (x-1)^2 + (y-1)^2 < (\frac{1}{4})^2\}$. The free boundary $\partial D_2(0)$ was the portion of the circle of radius $r = \frac{1}{4}$ lying in the unit square. The functions g_1 and g_2 were chosen to qualitatively model the ablation of a material occupying $D(t)$ and maintained at a given temperature α on the axes due to radiative input from a source at $x = 1, y = 1$. Specifically, we chose the source terms

$$g_1 = (1-x)/((1-x)^2 + (1-y)^2)^{3/2}, \quad g_2 = (1-y)/((1-x)^2 + (1-y)^2)^{3/2}$$

to denote an energy input on the free surface proportional to the inverse of the distance to the source times the cosine of the angle of incidence between the direction to the source and the respective coordinate axes.

In Fig. 1 we display the progress of the free boundary $\partial D_2(t)$ for the special boundary and initial conditions $\alpha = 1$ and $u_0 = 1$ on $\bar{D}(0)$, $u_0 = 0$ otherwise. The solution is now expected to be symmetric with respect to the line $y = x$. The code of [3] could handle the boundaries quite effectively. Going from a 40×40 grid on the unit square and a time step of $\Delta t = 0.2/100$ to the much coarser 20×20 grid and time step $\Delta t = 0.2/50$ resulted in less than 4% change in the position of the free surface while further refinements (up to 100×100 grids) left the position unchanged. In Figs. 1a and 1b we show the free surface every ten time step on the 40×40 and 20×20 grids. The irregular behavior near $y = 1$ should be ignored because the free surface $y = s_1(x, t)$ is plotted for fixed x values which in general do not coincide with the point of intersection between $s_1(x, t)$ and $y = 1$. The actual free boundaries are quite symmetric. For example, the computed values for the intersection with $x = 1$ and $y = 1$ in Fig. 1a are $(1, .4411)$ and $(.4483, 1)$ at $t = 0.2$. It is apparent from these results that the fractional step method could cope quite well with the free boundaries observed here. On the basis of the numerical experiments in [3] this was expected. Some comments on the run times: 50 time steps with the 20×20 grid required 22 secs, 100 time steps with the 40×40 grids required 178 secs, and 40 time steps with a 100×100 grid required 329 secs which amounts to roughly 10^{-3} secs per mesh point per time step.

The method of lines approach, (with y taken as the continuous variable) on the other hand, could not cope adequately with this problem. Admittedly, the asymmetry of the boundary conditions and the vertical tangent of the free surfaces at $y = 1$ made this method less attractive than the fractional step method. However, the method had worked quite well in the sample problems of [4] and [5] so that its poor performance here came as somewhat of a surprise. A representative result, somewhat analogous to that of Fig. 1a is shown in Fig. 2. Again, the shape of the free boundary deteriorates near $y = 1$ because its intersection with $y = 1$ is set to nearest mesh point on the left. Because of the coarse x -grid, the distortion is quite pronounced; however, the slow convergence associated with a more evenly spaced grid and the enormously long execution times present for large grids precluded an effective refinement of the above grid. A comparison between the answers of Fig. 1a and Fig. 2 shows discrepancies of the order of 12% on the line $x = 1$ and of 21% on the line $y = 1$ at $t = 0.2$.

These results indicate that the method of fractional steps is considerably faster and in certain geometries more accurate than the method of lines as presently implemented. Since the latter method applies under considerably weaker hypotheses the challenge is out to improve the method of lines, possibly through a finite element space discretization to make it competitive with the fractional step method.

REFERENCES

- [1] B. Budak, E. Sobol'eva, and A. Uspenskii, A difference method with coefficient smoothing for the solution of Stefan problems, *Zh. Vychisl. Mat. mat. Fiz.* 5 (1965), pp. 828-840.
- [2] G. H. Meyer, On a free interface problem for linear ordinary differential equations and the one phase Stefan problem, *Numer. Math.* 16 (1970), pp. 248-267.
- [3] G. H. Meyer, An alternating direction method for multi dimensional parabolic free surface problems, *Int. J. Num. Meth. Engng.* 11 (1977), pp. 741-752.
- [4] G. H. Meyer, The numerical solution of Stefan problems with front tracking and smoothing methods, *Appl. Math. Comp.*, to appear.
- [5] G. H. Meyer, An application of the method of lines to multi dimensional free boundary problems, *J. Inst. Math. Applcs.*, to appear.
- [6] G. H. Meyer, The method of lines for Poisson's equation with nonlinear boundary conditions, submitted for publication.
- [7] A. Samarskii and B. Moiseyenko, An economic continuous calculation scheme for the Stefan multi-dimensional problem, *Zh. Vychisl. Mat. mat. Fiz.* 5 (1965), pp. 816-827.
- [8] N. Yanenko, *The Method of Fractional Steps*, M. Holt, transl., Springer Verlag, Berlin, 1977.

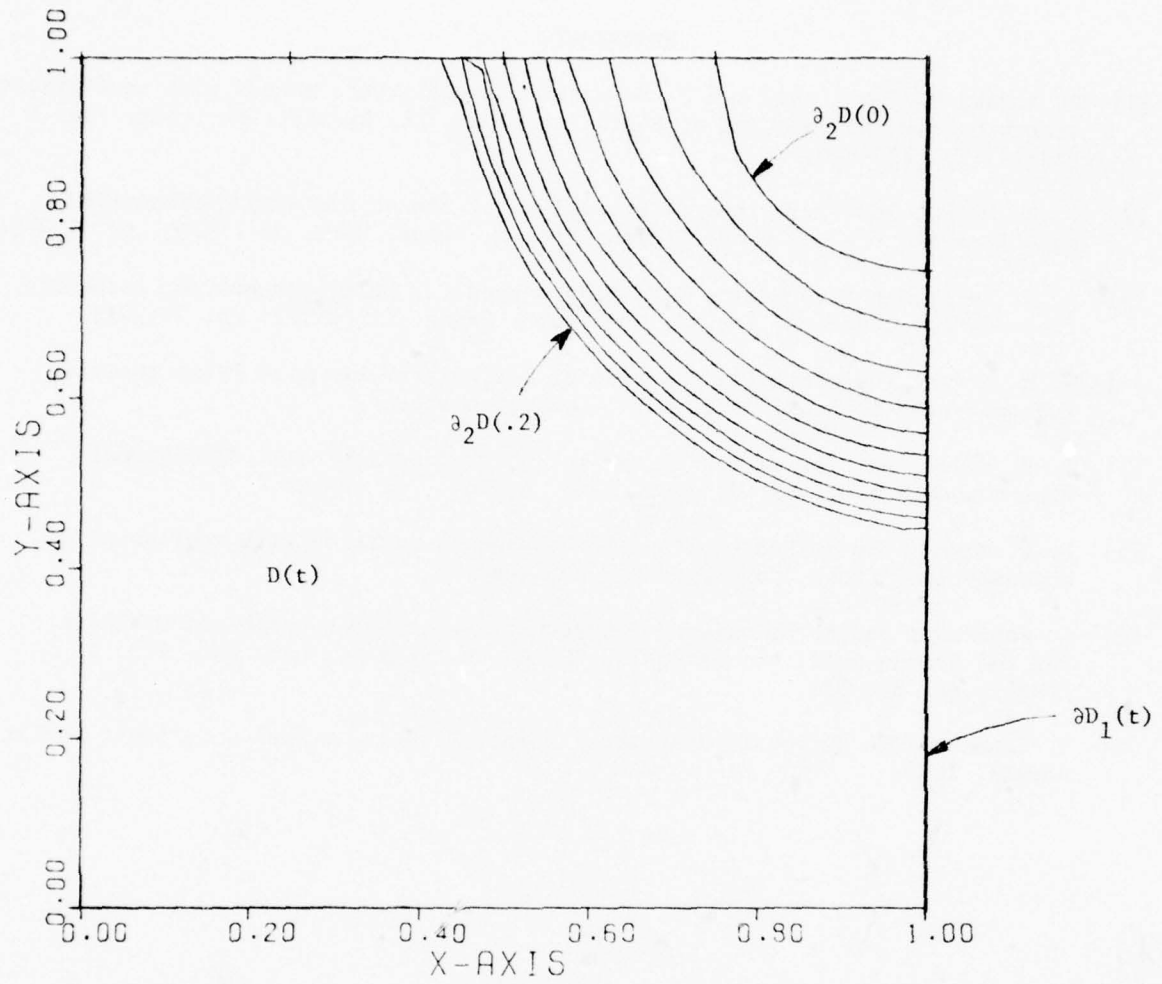


Fig. 1a. Free surfaces after every 10 time steps for the ablation problem (4.1-5). Fractional step method solution for $\Delta x = \Delta y = 1/40$; $\Delta t = 0.2/100$. Computing time on the CDC Cyber 74: 178 secs.

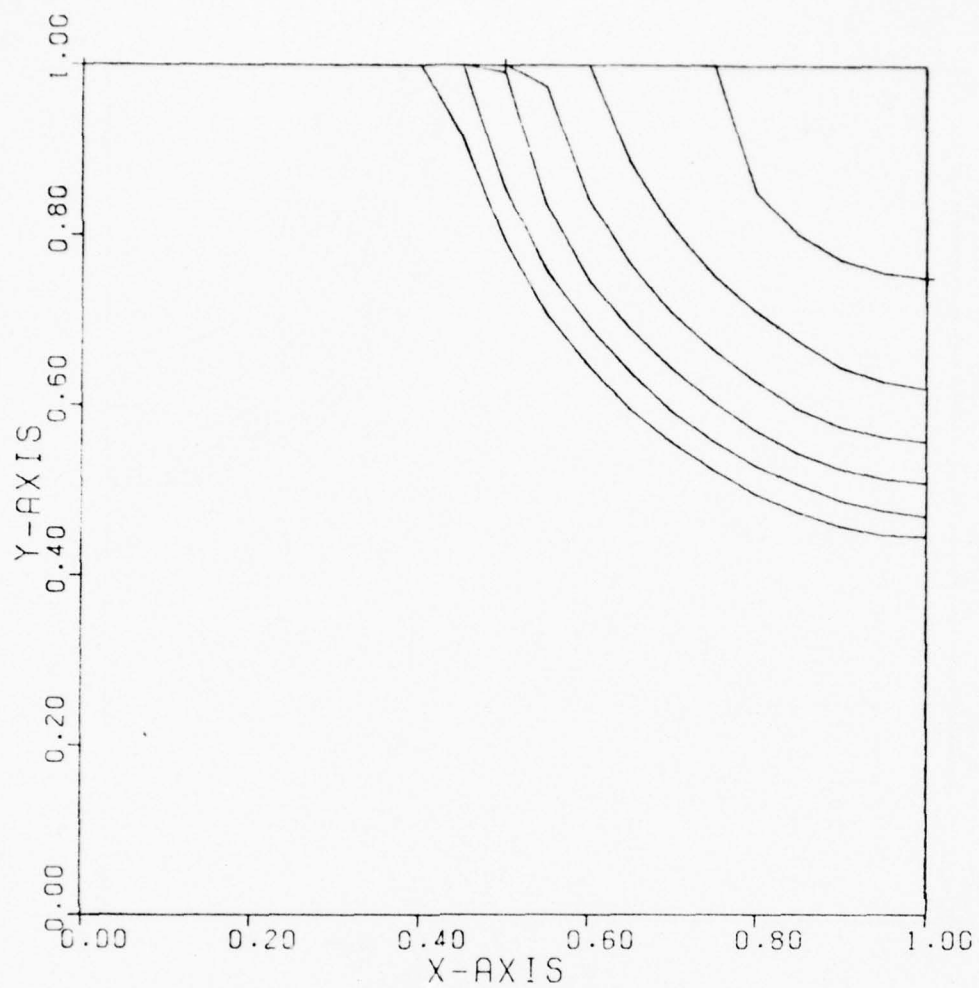


Fig. 1b. Free surfaces after every 10 time steps for the ablation problem (4.1-5). Fractional step method solution for $\Delta x = \Delta y = 1/20$; $\Delta t = 0.2/50$. Computing time on the Cyber 74: 22 secs.

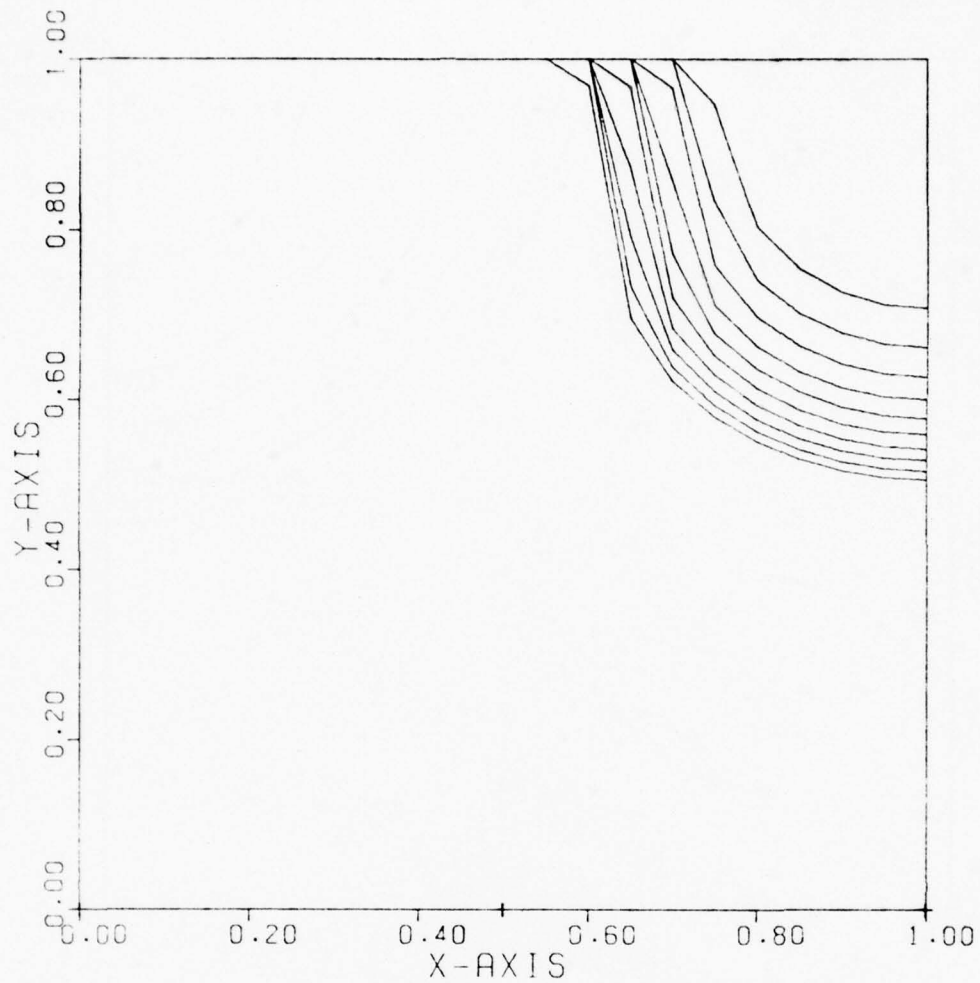


Fig. 2. Free surfaces after every 10 time steps for the ablation problem (4.1-5). Method of lines solution for $\Delta x = 1/20$, $\Delta y = 1/40$; $\Delta t = 0.2/100$. For a relaxation factor of $w = 1.4$ about 10 SOR cycles were required for each time step. The results were asymmetric and poor compared to the fractional step results. Computing time was an enormous 980 secs. The method of lines in its present implementation did not solve this problem satisfactorily.

ON THE NUMERICAL SOLUTION OF THE
POISSON EQUATION BY THE CAPACITANCE MATRIX METHOD

Arthur S. Shieh
Mathematics Research Center
University of Wisconsin, Madison
Madison, WI 53706

ABSTRACT. A discrete potential theory is developed for the capacitance matrix method that enables us to prove rapid convergence of conjugate gradient iterations in solving the capacitance matrix equation. Operation count of constant $n^2 \log n$ can be attained for second order accuracy schemes of the Neumann problem and high order accuracy schemes of the Dirichlet problem.

1. INTRODUCTION. The purpose of this paper is to present some theoretical and experimental aspects of efficient solvers for the linear systems of equations arising from finite differences discretization of the Dirichlet or the Neumann problem of the Poisson equation. The operation count of our algorithm is proportional to $n^2 \log n$, where $n = 1/h$. It is shown that high order accuracy can be achieved for the Dirichlet problem and second order accuracy can be achieved for the Neumann problem. The algorithm given here can be easily extended to the Helmholtz equation and to three dimensional problems.

2. CERTAIN RESULTS OF CLASSICAL POTENTIAL THEORY. We give here a brief summary of certain classical results that are useful for our purposes.

Let \mathcal{V} denote the potential resulting from a single layer charge distribution ρ on a smooth boundary curve $\partial\Omega$,

$$\mathcal{V}(x) = (1/\pi) \int_{\partial\Omega} \rho(\xi) \log r \, ds(\xi).$$

Here $x = (x_1, x_2)$, $\xi = (\xi_1, \xi_2)$ and $r^2 = (x_1 - \xi_1)^2 + (x_2 - \xi_2)^2$. Similarly, the potential \mathcal{W} of a dipole density on $\partial\Omega$ is defined by

$$\mathcal{W}(x) = (1/\pi) \int_{\partial\Omega} \mu(\xi) (\partial/\partial\nu_\xi) \log r \, ds(\xi).$$

We adopt the convention that the normal direction is toward the exterior of the region Ω in which we want to solve our problem. It can be shown (see e.g. [1] or [2]) that the Neumann and Dirichlet problems of Poisson's equation can be reduced to Fredholm integral equations of the second kind. For the interior Dirichlet problem, we make the Ansatz

$$u(x) = \mathcal{W}(x)$$

for the solution of

$$\Delta u = 0, \quad x \in \Omega$$

(2.1)

$$u = q, \quad x \in \partial\Omega.$$

The boundary condition is satisfied by choosing μ such that

$$\mu + (1/\pi) \int_{\partial\Omega} \mu (\partial/\partial\nu_{\xi}) \log r \, ds(\xi) = g.$$

This equation can be written as

$$(2.2) \quad (I + K)\mu = g$$

where K is the compact integral operator defined by the integral above.

For the interior Neumann problem, we make the Ansatz

$$u(x) = \mathcal{V}(x)$$

for the solution of

$$(2.3) \quad \begin{aligned} \Delta u &= 0, \quad x \in \Omega \\ \partial u / \partial \nu &= g, \quad x \in \partial\Omega. \end{aligned}$$

The boundary condition is satisfied by choosing ρ such that

$$(2.4) \quad (I - K^T)\rho = g,$$

where K is the same as in (2.1).

It is known that (2.2) has a unique solution for each g and the solution of (2.4) is unique up to an additive constant.

3. THE CAPACITANCE MATRIX METHOD. We are concerned with solving (2.1) or (2.3) by the finite difference method. Let R_h denote the uniform mesh on the plane.

Let $\Omega_h = \Omega \cap R_h$. Let A^* denote the coefficient matrix of our discrete problem on Ω_h . Let G denote the discrete analog of $\log r$ with respect to the five point formula for the discrete Laplacian. Let B denote the coefficient matrix for the discrete Laplacian; $BGv = v$ and Gv is well defined for any $n \times n$ vector v . We extend the domain of A^* to R_h by adjoining the rows of B on $R_h \setminus \Omega_h$ to A^* . The new matrix A thus formed differs with B by only m rows, where m = the number of irregular mesh points near the boundary. Note that $m \sim 1/h$. We can therefore write

$$A = B + UW^T$$

where U is the $n^2 \times m$ extension by zero matrix and W^T is an $m \times n^2$ matrix. U retains the value of any vector defined only on the irregular mesh points and extends it by zero to all mesh points. W^T describes how the boundary conditions are approximated. The restriction of the solution u of

$$(3.1) \quad Au = v.$$

to Ω_h is the solution for our original discrete problem provided that A is a reducible matrix with no coupling from Ω_h to $R_h \setminus \Omega_h$.

We now describe our algorithm for solving (3.1). In the Dirichlet case, we make the double layer Ansatz

$$u = Gv + GV\mu .$$

The m -vector μ is determined by solving

$$(3.2) \quad (I_m + W^T GV)\mu = -W^T Gv .$$

The $n^2 \times m$ matrix V transforms μ into a double layer charge distribution. Each column of V contains a discrete dipole of unit strength in the normal direction. See e.g. section 3 of [3] for details.

In the Newumann case, we make the single layer Ansatz

$$u = Gv + GU\rho .$$

The m -vector ρ should satisfy the equation

$$(3.3) \quad (I_m + W^T GU)\rho = -W^T Gv .$$

Both equations (3.2) and (3.3) are called capacitance matrix equations. The matrix on the left hand side of (3.2) or (3.3) is called the capacitance matrix which we denote by C . See also [4] for details.

We now describe our method of constructing W^T . In the Dirichlet case, we use the following schemes for interpolating the boundary conditions.

The first scheme which we shall call scheme I is as follows. Let $P \in \partial\Omega_h$, the set of irregular mesh points. Suppose that the local orientation of boundary is such that either W , the western neighbour of P , is in $R \setminus \Omega_h$ or both W and N , the northern neighbour of P is in $R \setminus \Omega_h$. We form the equation

$$4u(P) - u(E) - u(S) = h^2 f(P) + 2g(P) .$$

In the second scheme, called scheme II, we use an interpolation formula of second order accuracy for the Dirichlet data and obtain the equation

$$4u(P) - (1-\alpha_1) u(t) - (1-\alpha_2) u(s) = h^2 f(P) + (\alpha_3 + \alpha_4) g(P)$$

where $\alpha_1, \alpha_2, \alpha_3$ and α_4 are suitable Lagrangian coefficients. In the third scheme, which we shall call scheme H, we use a sixth order interpolation formula to interpolate the Dirichlet data. The equation at P will typically be in the form

$$4u(P) - \sum_{i=1}^5 \beta_i u(E_i) - \sum_{i=1}^5 \gamma_i u(S_i) = h^2 f(P) + \sum_{i=1}^{10} \delta_i g(P) ,$$

where E_i, S_i are suitably chosen mesh points very close to P and β_i, γ_i and δ_i arise from the Lagrangian coefficients. See e.g. [5] for details.

The last scheme is a highly artificial scheme not intended for practical use. It is, however, indispensable for theoretical purposes. Typically we form the equation

$$(5 + \alpha) u(P) - u(E) - u(S) - u(N) - \alpha u(P') = h^2 f(P) + 2g(P).$$

The parameter α is to be determined as described in section 5. P' is a suitably chosen point sufficiently close to P . We shall refer to this scheme as scheme X.

In the Neumann case, we use the following schemes. The first scheme which we refer to as scheme I.N is constructed as follows. Let α be the angle that the normal through P intersecting the boundary at P^* makes with the x-axis. We approximate $[\partial u / \partial \nu](P^*)$ by $[u(W) - (1 - \tan \alpha) u(P) - (\tan \alpha) u(S)] / h \sec \alpha$. We then eliminate $u(W)$ from the five-point formula to obtain

$$(3 + \tan \alpha) u(P) - u(E) - u(N) - (1 + \tan \alpha) u(S) = h^2 f(P) + h \sec \alpha g(P^*).$$

The second scheme which we call scheme II is much more elaborate. The normal derivative at P^* is approximated as a linear combination of $u(W)$, $u(S')$ and $u(E')$. Here S' and E' are respectively the points on the normal through W intersecting the mesh lines through S and E . We then interpolate $u(S')$ by a linear combination of $u(N)$, $u(P)$ and $u(S)$ and do a similar interpolation for $u(E')$. We then eliminate $u(W)$ from the five-point formula and obtain a six-point formula for the Laplacian and the boundary condition. See e.g. [6] for details. We remark that this is not the most compact scheme possible. Bramble and Hubbard in [7] obtained a positivity scheme using only three mesh points while scheme II.N does not even have positivity. It does, however, use values at points very close to P and we can obtain convergence proof using the discrete potential theory sketched in section 5. See [6] for details.

Finally we remark that as far as the Poisson equations is concerned, we can first find the values of u at $\partial \Omega_h$ using the integral equation method and then back solve via the capacitance matrix method. See e.g. [6]. The scheme II.N, however, has the advantage to be easily extendable to the Helmholtz equation and to the three dimensional problems.

4. NUMERICAL PROCEDURES AND OPERATION COUNTS. In the Dirichlet case, we use the discrete Green's function on the entire plane as our G_2 in equations (3.2)-(3.3). It is shown in [3] that $G(r,s) = (1/4\pi) \log(r^2+s^2) + c_1/(r^2+s^2) + c_2 r^2 s^2 / (r^2+s^2)^3 + R_1$, where $|R_1| \leq C_3 / (r^2+s^2)^2$, c_1, c_2 and c_3 are known positive constants. We can therefore generate G by one call of a fact solver using distant values of G as the Dirichlet data.

Because of translational invariance, this is all we need to set up the capacitance matrix C . Equation (3.2) is then solved by the conjugate gradient method. We shall see in section 5 that because of the favorable distribution of the singular values of C , the convergence is essentially independent of h . Typically, only four or five iterations are needed to achieve an accuracy of 10^{-3} . The operation counts are therefore approximately $10m^2$ (Note that $C^T C u = C^T b$ is actually solved for the conjugate gradient routine).

The capacitance matrix equation can be simplified to $C u = U^T v$ if $v = U U^T v$. This will be the case if we solve the Laplace equation instead of the more general

Poisson and if we make the Ansatz $u = GV\mu$. Because $V\mu$ is a sparse vector, the boundary values of u on the sides of a rectangle containing Ω can be computed at a cost of constant n^2 . Therefore we can backsolve for u by only one call of fast Poisson solver. The total operation counts in this case is just approximately $5n^2 \log n + 80n^2$. If v is not equal to $UU^T v$, we must first compute $W^T Gv$. It is shown in [3] that $W^T Gv$, Gv and $GV\mu$ can be computed at approximately the cost of two calls of fast Poisson solvers. The total operation counts are estimated to be $10n^2 \log n + 120n^2$.

In the Neumann case, we have used the iterative imbedding technique first used by J. A. George in [8]. We no longer require G to be translational invariant. The vectors Cp are computed as follows. First compute Up . Solve $By = Up$. Then compute $p + W^T y$. We note that Up is a sparse vector and the values of y are only needed in a neighbourhood of $\partial\Omega_h$. Therefore $W^T y$ can be computed at a cost of approximately $8n^2$ using the discrete Fourier transform methods studied in [9] or [10]. The total operation count is therefore approximately $160n^2$.

5. DISCRETE POTENTIAL THEORY. In the Dirichlet case, we must show that C is nonsingular for at least sufficiently small mesh sizes. In the numerical experiments C is uniformly well conditioned (see also [11]) if the double layer Ansatz is used. It is, however, much more difficult to prove rigorously that C must be nonsingular. The proof given in [11] can only be modified to prove the nonsingularity of C for the single layer case. In the following, we sketch two proofs, one for special cases and the other for general regions.

DEFINITION. The near diagonal part B_h of C is defined to be the matrix that satisfies the following. $B_h(P, Q) = C(P, Q)$ if $d(P, Q) \leq \sqrt{h}$, $B_h(P, Q) = 0$ if otherwise.

DEFINITION. The off diagonal part K_h of C is defined by $K_h = C - B_h$.

It is easy to see that the K_h of suitably scaled C are formal approximation to the compact operator K in equation (2.3). Choose operators from $\mathcal{C} \rightarrow \mathcal{C}$ such that K_h have the same nonzero eigenvalues as K_h . Here \mathcal{C} is the Banach space of continuous functions on $[0, 1]$ with the sup norm. It is shown in [3] that the following holds.

THEOREM 1. $B_h + B_h^T \geq \alpha I$; $\alpha \geq 1$ for scheme I, $\alpha \geq 0.4$ for scheme II. $\{K_m\}$ is collectively compact on \mathcal{C} , $K_m x \rightarrow Kx$ for each $x \in \mathcal{C}$.

It can then be shown from the theory of collectively compact operators that given $\epsilon > 0$, $K_h + K_h^T \geq K + K^T - \epsilon$ for sufficiently small h .

DEFINITION. $\Omega \in \mathfrak{F}_\beta$ if $K + K^T > -\beta I$. Here K is the compact operator associated with Ω by the line integral in (2.2).

The foregoing discussion then easily lead to the following result.

THEOREM 2. Let $\Omega \in \mathfrak{F}_\beta$, $\beta = 1$ (or 0.4) if scheme I (or II) is used. Then C is uniformly invertible in the spectral norm.

It is known that \mathfrak{F}_β is nonempty for any $\beta \geq 0$. All ellipses with $(a-b)/(a+b) = \beta$ are in \mathfrak{F}_β , where a, b are the half axes of the ellipse. See e.g. [3].

We next establish that C is nonsingular with its smallest singular value larger than $1/[K(A)]^2$ where $K(A)$ denotes the condition number of A for at least sufficiently small mesh sizes. Let $\|C\|$ denote the maximum norm of a matrix C . It is shown in [3] that $\|C^{-1}\| \leq 1/[K(A)]^2$ if there exists a scheme of interpolating boundary conditions such that the corresponding C is uniformly invertible in the maximum norm. It can be shown (see e.g. [12]) that suitably chosen α 's will make the row sums of B_h one while preserving the diagonal dominance of B_h . If we now form B_m from B_h in the same way as K_m are formed from K_h , then the following holds.

LEMMA 1. If scheme X is used, then $B_m x \rightarrow x$ for each $x \in \mathcal{C}$. $\|B_m^{-1}\|_\infty \leq \text{constant}$ and $B_m^{-1} x \rightarrow x$ for each $x \in \mathcal{C}$. We can then prove the following theorem very easily using some basic results from the theory of collectively compact operators. See [12] for details of the proof. See also [13] for a good account of the theory.

THEOREM 3. Suppose $B_m x \rightarrow x$ and $\|B_m^{-1}\|_\infty \leq \text{constant}$. If $\{K_m\}$ is collectively compact and $K_m x \rightarrow x$ for each $x \in \mathcal{C}$, then $\|B_m + K_m\|^{-1} \leq \text{constant}$ iff $I + K$ is invertible.

It is easy to see that the above theorem implies that C for scheme X is uniformly invertible in the maximum norm. This proves our claim that C is nonsingular in general. The following theorem guarantees the rapid convergence of the conjugate gradient iterations.

THEOREM 4. Suppose $C = B_h + K_h$, where $\|B_h^{-1}\|_\infty \leq \text{constant}$ and the singular values of K_h cluster around that of a compact operator K . If in addition C is nonsingular with its smallest singular value $\geq h^{-p}$, $p = \text{positive constant}$, then the number of conjugate gradient iterations needed to reduce the error $E(x)$ to a given accuracy does not exceed constant times $\log m$. Here $E(x) = (1/2) (x - x^*)^T Q (x - x^*)$, where $Q = C^T C$ and x^* satisfies $Qx^* = C^T b$.

PROOF. See e.g. [3].

A similar theorem also holds for the Neumann case, see e.g. [4]. The theory for the second order case is much more complicated and we again have to go back to classical potential theory and use a theorem similar to theorem 3. It is, however, not necessary to introduce an artificial scheme X . Instead, an artificial splitting of the matrix U has to be introduced. See e.g. [6] for details.

5. NUMERICAL EXPERIMENTS. We have found that the convergence rate in the Dirichlet and Neumann cases vary very little when the domains and mesh sizes are changed. The following table represents a typical situation. The Neumann problem is solved on ellipses with $b/a = \gamma$, where a, b are the half axes. The test solution is $\sin(x+y)$. The right hand side is therefore $2h^2 \sin(x+y)$ on regular mesh points and $2 \cos(x+y) (\pm 1.0 \pm \tan(\alpha)) \cdot h + 2h^2 \sin(x+y)$ on irregular mesh points. In Table I, $E(Vu) = \max\{|\delta_x(u_h - u)|, |\delta_y(u_h - u)|\}$, where δ_x and δ_y are respectively the difference quotients in x and y coordinate directions. The norm of C.G. residual is computed by dividing the L_2 norm of the conjugate gradient residuals by the square root of estimated mesh points inside Ω .

TABLE I

$$\Delta u = 2 \sin(x+y)$$

Scheme I.N. ($h = 1/48$)

No. of iterations	No. of irregular mesh points	γ	Norm of C.G. Residual	$E(u)$
2	108	1.0	.2803274-03	1.0-02
3	108	1.0	.4790281-04	1.0-02
2	92	0.7	.3755522-03	2.5-02
3	92	0.7	.1263798-03	2.0-02
2	84	0.5	.6496120-03	6.0-02
3	84	0.5	.3265076-03	4.0-02

We remark that it is very important that the right hand side of the continuous problem is consistent and that the right hand side of the capacitance matrix should be consistent whenever the same holds for the discrete problem. The latter is either guaranteed by discrete potential theory if we split up the matrix U or it is automatically satisfied if we do not split up U as in all practical applications. See e.g. [4] for details. In spite of this, it is, however, not necessary to check for consistency in the original discrete problem. In fact, the solution produced by our algorithm is not sensitive to small changes in the right hand side of the discrete problem.

In the Dirichlet case, we can obtain results of high order accuracy if scheme H is used. Deferred correction methods or Richardson extrapolation can be used to obtain sixth order accuracy results. See [5] for details. The table II is taken from section of [5]. Similar results of fourth order accuracy obtained by conjugate gradient method with operation counts proportional to $n^2 \log n$ can be found in [12].

TABLE II

$$-\Delta u = 2 \sin(x+y)$$

$$u(x) = \sin(x+y)$$

Scheme H

Correction	0	1	2
$\epsilon_{\infty}, k = 6$	$1.9 \cdot 10^{-5}$	$1.0 \cdot 10^{-9}$	$5.6 \cdot 10^{-12}$
$\epsilon_2, k = 6$	$1.0 \cdot 10^{-5}$	$5.4 \cdot 10^{-10}$	$3.4 \cdot 10^{-12}$

L_2 and maximum-norm errors for Dirichlet problem on a circle

6. BIBLIOGRAPHY

- [1] P. R. Garabedian, Partial Differential Equations, Wiley, 1964.
- [2] I. G. Petrovsky, Partial Differential Equations, Interscience, 1954.
- [3] A. Shieh, Fast Poisson solvers for general regions, I. The Dirichlet problem, to appear.
- [4] A. Shieh, On the convergence of the conjugate gradient method for singular capacitance matrix equations, MRC TSR #1730.
- [5] V. Pereyra, W. Proskurowski and O. Widlund, High order fast Laplace solves for the Dirichlet problem on general regions, Math. Comp., 1977.
- [6] Shieh, Fast Poisson solves for general regions, III. Neumann problem, to appear.
- [7] J. H. Bramble and B. E. Hubbard, The Neumann problem for Poisson's equation, SIAM J. Numer. Anal., Ser. B, 2 (1965), 1-14.
- [8] J. A. George, The use of direct methods for the solution of the discrete Poisson equation, C. S. Dept. Report 159, Stanford University, 1970.
- [9] A. Banegas, Fast Poisson solvers for sparse data, to appear.
- [10] W. Proskurowski, Lawrence Berkeley Report, to appear.
- [11] J. W. Proskurowski and O. Widlund, On the numerical solution of Helmholtz's equation by the capacitance matrix method, Math. Comp., 1976.
- [12] A. Shieh, Fast Poisson solver for general regions, II. The Dirichlet problem (cont'd), to appear.
- [13] P. M. Anselonne, Collectively Compact Operator Approximation Theory, Prentice-Hall, N.J.

A POWER SERIES SOLUTION OF A HARMONIC MIXED BOUNDARY VALUE PROBLEM

J. Barkley Rosser* and N. Papamichael**

ABSTRACT. The first twenty coefficients for a boundary value problem are approximated. This gives a useful conformal transformation.

1. Definition of the coefficients.

We consider a rectangle R with corners $(\pm 1, 0)$ and $(\pm 1, 1)$.

Let $u(x, y)$ be the function continuous on and inside the rectangle such that the second partial derivatives are continuous inside the rectangle, the first partial derivatives are continuous on and inside the rectangle except at the point $(0, 0)$,

$$(1.1) \quad \nabla^2 u(x, y) = 0$$

for (x, y) inside the rectangle, and

$$(1.2) \quad u(x, 0) = 0 \quad -1 \leq x \leq 0$$

$$(1.3) \quad u_y(x, 0) = 0 \quad 0 < x < 1$$

$$(1.4) \quad u_y(x, 1) = 0 \quad -1 < x < 1$$

$$(1.5) \quad u_x(-1, y) = 0 \quad 0 < y < 1$$

$$(1.6) \quad u(1, y) = 1 \quad 0 \leq y \leq 1.$$

*

Mathematics Research Center, University of Wisconsin, and
Department of Mathematics, Brunel University.

Sponsored in part by the United States Army under Contract
No. DA-31-124-ARO-D-462 and in part by the Science Research Council
under grant B/RG 4121 at Brunel University.

**

Department of Mathematics, Brunel University.

The function $500 + 500 u(x,y)$ was studied in Problem 2 on p.660 of Whiteman and Papamichael [1]. They cite a considerable number of papers where functions as closely related to $u(x,y)$ as theirs have been studied.

Let us put

$$(1.7) \quad z = x + iy.$$

Then $u(x,y)$ is the real part of a function $u(z)$ which is analytic inside the rectangle.

We will show that there are real constants a_n such that inside R

$$(1.8) \quad u(z) = \sum_{n=0}^{\infty} a_n z^{n + \frac{1}{2}}.$$

Indeed, the series on the right of (1.8) has radius of convergence 2.

Thus, we conclude that

$$(1.9) \quad u(r \cos \theta, r \sin \theta) = \sum_{n=0}^{\infty} a_n r^{n + \frac{1}{2}} \cos(n + \frac{1}{2})\theta$$

for positive values of r and for $0 < \theta < \pi$ such that the point $(r \cos \theta, r \sin \theta)$ is inside R .

In the present report a procedure for determining the coefficients a_n in (1.8) is developed and is used to calculate numerical approximations \tilde{a}_n to a_n , for $0 \leq n \leq 19$. Approximations $\tilde{u}(x,y)$ to the solution $u(x,y)$ of problem (1.1)-(1.6) are then obtained, for various values of $(x,y) = (r \cos \theta, r \sin \theta)$ in R , from

$$\tilde{u}(x,y) = \tilde{u}(r \cos \theta, r \sin \theta) = \sum_{n=0}^{19} \tilde{a}_n r^{n + \frac{1}{2}} \cos(n + \frac{1}{2})\theta.$$

2. Some proofs.

The first order of business is to prove the various assertions made in Section 1. We shall make repeated use of a "reflection principle". Our first step is to "reflect" $u(x,y)$ relative to the x -axis by defining

$$(2.1) \quad v(x,y) = \begin{cases} u(x,y) & 0 \leq y \leq 1 \\ u(x,-y) & -1 \leq y \leq 0. \end{cases}$$

It is easily shown (see our later discussion of \bar{w}) that v is harmonic inside the square with corners $(\pm 1, \pm 1)$ except along the "slit" defined by $y = 0$, $-1 \leq x \leq 0$; along this "slit" it is continuous and equal to 0, but $v_y(x,y)$ is discontinuous across the "slit". On the right side of the square, $v = 1$, and on the other three sides the normal derivative is zero. As v is merely a continuation of u across the x -axis, we shall call it u .

By "reflection" relative to the lines $y = 1$ and $y = -1$, we can continue u into the rectangle with corners $(\pm 1, \pm 3)$; it will be harmonic inside the rectangle except along three "slits". By "reflection" relative to the line $x = -1$, we can continue u into the rectangle with corners $(-3, \pm 3)$ and $(1, \pm 3)$; again, it is harmonic except along three "slits". These three "slits" are the three straight line segments extending respectively from $(-2, 2)$ to $(0, 2)$, from $(-2, 0)$ to $(0, 0)$, and from $(-2, -2)$ to $(0, -2)$.

Finally, we define $w(x,y)$ by

$$(2.2) \quad w(x,y) = \begin{cases} u(x,y) & -3 \leq x \leq 1 \\ 2 - u(2-x,y) & 1 \leq x \leq 5. \end{cases}$$

To show that w is harmonic inside the large rectangle with corners

$(-3, \pm 3)$ and $(5, \pm 3)$ except along six "slits", we reason as follows. Take $\bar{w}(x,y)$ to be the function which is harmonic inside this large rectangle except along the six "slits", is zero on both vertical sides of the rectangle, has zero normal derivative along the top and bottom of the rectangle, is -1 along each of the three "slits" on the left, which go from $(-2,2)$ to $(0,2)$, from $(-2,0)$ to $(0,0)$, and from $(-2,-2)$ to $(0,-2)$ respectively, and is $+1$ along each of the three "slits" on the right, which go from $(2,2)$ to $(4,2)$, from $(2,0)$ to $(4,0)$, and from $(2,-2)$ to $(4,-2)$ respectively.

To show that there is such a $\bar{w}(x,y)$, we appeal to Theorem I.25 on p.31 of Tsuji [7]. Indeed, let us define the "Schottky prolongation" of our rectangle as at the bottom of p.31 of Tsuji [7]. Specifically, let us take a Riemann surface of two sheets, each a rectangle covering our large rectangle, fastened together along their top and bottom edges. We write (\bar{x}, \bar{y}) , $-3 \leq \bar{x} \leq 5$, $-3 \leq \bar{y} \leq 3$, to denote a point on the top sheet and (x, y) correspondingly to denote a point on the bottom sheet. Then, as in (i) of the proof of Theorem I.25, we seek a function $\bar{w}(x,y)$ such that

$$\bar{w}(\bar{x}, \bar{y}) = \bar{w}(x, y) .$$

Accordingly we are reduced to a strictly Dirichlet problem, since the upper and lower edges of the rectangle are no longer boundaries. Hence, it suffices to appeal to the existence theorem, Theorem I.12 on p.8 of Tsuji [7], for the Dirichlet problem. Similarly, using the same "Schottky prolongation" of our large rectangle, we can reduce the uniqueness of $\bar{w}(x,y)$ to uniqueness for a strictly Dirichlet case. Specifically, let $\bar{w}(x,y)$ be a solution on the rectangle, whose uniqueness

is to be deduced. Define \bar{w} for the Riemann surface by

$$\bar{w}(\bar{x}, \bar{y}) = \bar{w}(x, y) = \bar{w}(\underline{x}, \underline{y}).$$

Then, since $\bar{w}(x, y)$ and its derivatives of first order have continuous limits on the top and bottom edges of the rectangle, the same is true of $\bar{w}(\bar{x}, \bar{y})$ and $\bar{w}(\underline{x}, \underline{y})$. Also, $\bar{w}(\bar{x}, \bar{y})$ and $\bar{w}(\underline{x}, \underline{y})$ are equal and have equal normal derivatives (namely zero) along the top and bottom edges of the rectangle. So, by Theorem VI on p.261 of Kellogg [8], \bar{w} is a harmonic function on the Riemann surface. Thus we are reduced to a strictly Dirichlet case, and uniqueness follows from the generalized principle of the maximum, Theorem I.2 on p.2 of Tsuji [7].

There are numerous errors in Tsuji [7]. Most are easily rectified by an attentive reader. We should note that in his proof of Theorem I.12 (existence), Tsuji uses a stronger version of Theorem I.2 (maximum principle) than he states or proves. In Theorem I.2, the statement of (i) should be strengthened to:

(i) If $\phi(z)$ is subharmonic in D and for each $\zeta \in \Gamma$,

$$\overline{\lim}_{z \rightarrow \zeta} \phi(z) \leq M, \quad z \in D,$$

then $\phi(z) \leq M$ in D .

Statements (ii) and (iii) should similarly be strengthened.

To prove the strengthened version of (i), suppose that there is an $\epsilon > 0$ such that there are points of D for which $\phi(z) \geq M + \epsilon$.

If there are such points, the totality of them can be shown to constitute a closed region, completely interior to D , because $\phi(z)$ is continuous in D , since that is part of the definition of "subharmonic". But then, in this closed region, $\phi(z)$ must attain

its maximum value, which will be the maximum value of $\phi(z)$ in D . Then the rest of the proof proceeds very similarly to the proof given in Tsuji [7] for the weaker version of (i).

It happens that in his proof of Theorem I.12 (existence), Tsuji uses the denumerable axiom of choice. As this axiom seems indispensable for the theory of Lebesgue integration, most analysts could not reasonably object to Tsuji's proof. However, for our present purpose we can dispense with the axiom of choice. For the benefit of any purists among our readers we shall indicate how to do this.

If $u(x,y)$ exists, and we define $w(x,y)$ by (2.2) then by Theorem VI on p.261 of Kellogg [8] $w(x,y)$ is harmonic. If we then define

$$\bar{w}(x,y) = w(x,y) - 1,$$

it will be seen to have the required properties. So we are reduced to the question of the existence of $w(x,y)$. If we follow this back, we are reduced finally to the question of the existence of a $u(x,y)$ satisfying (1.1) through (1.6). But this existence can be concluded by means of the conformal transformation defined in Whiteman and Papamichael [1].

Thus, for the case at hand, we do not need to invoke Theorem I.12, the existence theorem of Tsuji [7]. The maximum principle, Theorem I.2, from which we can deduce uniqueness, is proved without using the axiom of choice.

It will be noted that both $\bar{w}(2-x,y)$ and $-\bar{w}(x,y)$ are harmonic inside the large rectangle with corners $(-3, \pm 3)$ and $(5, \pm 3)$ except along the six "slits", and both satisfy the same boundary conditions around the boundary of the rectangle, and along each of the "slits". So we conclude that

$$(2.3) \quad \bar{w}(2-x,y) = -\bar{w}(x,y).$$

From this, we infer that

$$(2.4) \quad \bar{w}(1,y) = 0.$$

So, inside the rectangle with corners $(-3, \pm 3)$ and $(1, \pm 3)$, $u(x,y)$ and $1 + \bar{w}(x,y)$ are both harmonic except along the three "slits", and both satisfy the same boundary conditions around the boundary of the rectangle, and along the "slits". So, inside this rectangle we have

$$(2.5) \quad u(x,y) = 1 + \bar{w}(x,y).$$

Then by (2.2) and (2.3), we conclude that

$$w(x,y) = 1 + \bar{w}(x,y)$$

holds in the large rectangle. So $w(x,y)$ is harmonic, as claimed.

We may think of w as continuing u into the large rectangle.

We now restrict attention to the circle of radius 2 with center at the origin. Inside this circle u is harmonic except along the "slit" defined by $y = 0$, $-2 \leq x \leq 0$; along this "slit" it is continuous and equal to 0.

Recall (1.7), in which we took

$$z = x + iy.$$

Then $u(x,y)$ is the real part of a function $u(z)$ which is analytic inside the circle except along the "slit".

By adding a constant if need be, we can determine that $u(0) = 0$.

As $u_y(x,y) = 0$ along the positive real axis, we conclude by the Cauchy-Riemann differential equations that $u(z)$ is real along the positive

real axis. So

$$(2.6) \quad u(\bar{z}) = \overline{u(z)} .$$

We transform from the z -plane to the ζ -plane by the transformation

$$(2.7) \quad z = \zeta^2 .$$

The "slit" goes into the part of the imaginary ζ -axis from $-i\sqrt{2}$ to $i\sqrt{2}$. The rest of the interior of the circle goes into the interior of a semi-circle of radius $\sqrt{2}$ with center at the origin lying to the right of the imaginary ζ -axis.

Inside the semi-circle, we define an analytic function $v(\zeta)$ by

$$(2.8) \quad v(\zeta) = u(\zeta^2) .$$

By (2.6) we have

$$(2.9) \quad v(\bar{\zeta}) = \overline{v(\zeta)} .$$

Along the imaginary ζ -axis from $-i\sqrt{2}$ to $i\sqrt{2}$, we have by (2.8) that the real part of $v(z)$ is zero. Then by (2.9) we have

$$(2.10) \quad v(-\zeta) = -v(\zeta)$$

along that part of the imaginary axis. We define $v(\zeta)$ by (2.10) for ζ in the left half of the circle of radius $\sqrt{2}$ with center at the origin. It is easily verified that the real part of $v(\zeta)$ is harmonic inside the entire circle; compare our earlier reasoning concerning \bar{w} . By the Cauchy-Riemann differential equations and the fact that $v(z)$ is continuous across the imaginary axis by (2.10),

we conclude that $v(\zeta)$ is analytic in the entire circle. Thus, it can be expanded in a power series

$$(2.11) \quad v(\zeta) = \sum_{n=0}^{\infty} b_n \zeta^n$$

with radius of convergence at least $\sqrt{2}$.

By (2.9), we conclude that the b_n are all real. By (2.10), b_n is zero when n is even. Thus, by (2.7) and (2.8) there are real a_n such that

$$(2.12) \quad u(z) = \sum_{n=0}^{\infty} a_n z^{n + \frac{1}{2}}.$$

This is (1.8), and by taking the real part of both sides we conclude that (1.9) holds, namely

$$(2.13) \quad u(r \cos \theta, r \sin \theta) = \sum_{n=0}^{\infty} a_n r^{n + \frac{1}{2}} \cos(n + \frac{1}{2})\theta.$$

We see that (2.12) has a radius of convergence at least as large as 2. However, the radius of convergence cannot be greater than 2. For suppose the radius of convergence is $R > 2$. Then (2.13) would be valid for $0 \leq r < R$. Then we could infer that $u_y(x, y)$ is continuous across part of the "slit" defined by $y = 0$, $2 \leq x \leq 4$; the latter is not the case. So the series (2.12) has radius of convergence 2.

The purpose of the present report is to explain a procedure for calculating the a_n , and to present approximations for a number of them.

One can find that series such as (2.12) are developed in Wasow [2]. It happens that the series developed there are only shown to be asymptotic. However, this does not exclude the possibility that in certain cases, such as the one we are considering, the series is actually convergent.

3. Some algorithmic considerations.

In order to calculate the a_n , it is desirable to be able to manipulate power series on a computer. We shall present some algorithms for this purpose.

Suppose

$$(3.1) \quad F(x) = \sum_{n=1}^{\infty} F_n x^n.$$

Define $F_n^{(j)}$ as the coefficient of x^{n+j-1} in $(F(x))^j$. That is,

$$(3.2) \quad (F(x))^j = \sum_{n=1}^{\infty} F_n^{(j)} x^{n+j-1}.$$

We clearly have

$$(3.3) \quad F_n^{(1)} = F_n,$$

$$(3.4) \quad F_n^{(j+1)} = \sum_{r=1}^n F_r F_{n+1-r}^{(j)}.$$

If we know F_n for $1 \leq n \leq N$, then $F_n^{(j)}$ can be calculated for as large a j as desired and for $1 \leq n \leq N$ by means of (3.3) and (3.4).

If only the values of $F_n^{(J)}$ for $1 \leq n \leq N$ are desired, one can get them without calculating $F_n^{(j)}$ for $1 < j < J$ by the following stratagem of Nijenhuis and Wilf [9]. By (3.2),

$$J \log F(x) = \log \sum_{n=1}^{\infty} F_n^{(J)} x^{n+J-1}.$$

If we differentiate both sides, and clear fractions, we get

$$J \sum_{n=1}^{\infty} F_n^{(J)} x^{n+J-1} \sum_{k=1}^{\infty} k F_k x^{k-1} = \sum_{n=1}^{\infty} (n+J-1) F_n^{(J)} x^{n+J-2} \sum_{k=1}^{\infty} F_k x^k.$$

If we equate the coefficients of x^{M+J} on both sides we get

$$J \sum_{n=1}^{M+1} F_n^{(J)} (M+2-n) F_{M+2-n} = \sum_{n=1}^{M+1} (n+J-1) F_n^{(J)} F_{M+2-n}.$$

So, for $M \geq 1$, we have

$$MF_{M+1}^{(J)} F_1 = \sum_{n=1}^M (J(M+1-n)-n+1) F_n^{(J)} F_{M+2-n}.$$

If $F_1 \neq 0$, this can be used to calculate $F_n^{(J)}$ recursively, beginning with

$$F_1^{(J)} = (F_1)^J.$$

Note that J need not be an integer.

Clearly, if

$$(3.5) \quad G(x) = \sum_{n=1}^{\infty} G_n x^{n-1},$$

then

$$xG(x) = \sum_{n=1}^{\infty} G_n x^n.$$

So, if we define $G_n^{(j)}$ by

$$(3.6) \quad G_n^{(1)} = G_n$$

$$(3.7) \quad G_n^{(j+1)} = \sum_{r=1}^n G_r G_{n+1-r}^{(j)},$$

we will have

$$(xG(x))^j = \sum_{n=1}^{\infty} G_n^{(j)} x^{n+j-1},$$

so that

$$(3.8) \quad (G(x))^j = \sum_{n=1}^{\infty} G_n^{(j)} x^{n-1}.$$

We realize that the treatment of $G(x)$ is mathematically unorthodox, but we have adopted it because, in writing FORTRAN programs, it is convenient to avoid using 0 as a subscript. Referring back to (2.12), let us define

$$(3.9) \quad A_n = a_{n-1} \quad 1 \leq n.$$

Then we have by (2.12)

$$(3.10) \quad u(z)\sqrt{z} = \sum_{n=1}^{\infty} A_n z^n.$$

Then

$$(3.11) \quad (u(z))^j = \sum_{n=1}^{\infty} A_n^{(j)} z^{n-1+\frac{1}{2}j},$$

where we define $A_n^{(j)}$ analogously to $F_n^{(j)}$ and $G_n^{(j)}$.

Let us have

$$(3.12) \quad B(x) = \sum_{n=0}^{\infty} B_n x^n,$$

with $B_0 \neq 0$. To get the coefficients of the power series expansion for $(B(x))^{-1}$, we proceed as follows. We have

$$(3.13) \quad B(x) = B_0 (1 + F(x))$$

where $F(x)$ is given by (3.1) with

$$(3.14) \quad F_n = \frac{B_n}{B_0}, \quad 1 \leq n.$$

Then, if we define C_n by

$$(3.15) \quad \frac{1}{B(x)} = \frac{1}{B_0} \left\{ 1 + \sum_{n=1}^{\infty} C_n x^n \right\},$$

we have

$$(3.16) \quad 1 = \{1 + F(x)\} \left\{ 1 + \sum_{n=1}^{\infty} C_n x^n \right\}.$$

Equating the coefficients of x^{M+1} on both sides gives the recursion

$$(3.17) \quad C_{M+1} = -F_{M+1} - \sum_{n=1}^M C_n F_{M+1-n}, \quad 1 \leq M.$$

From this we have $C_1 = -F_1$. If C_1, C_2, \dots, C_M have been calculated and stored in the computer then, by (3.17), C_{M+1} can be calculated and stored in the computer.

Consider next the question of a power series expansion for $(B(x))^{1/J}$, where J is greater than unity and need not be an integer. By (3.13), we have

$$(3.18) \quad (B(x))^{1/J} = B_0^{1/J} (1 + F(x))^{1/J}.$$

We define D_n by

$$(3.19) \quad 1 + \sum_{n=1}^{\infty} D_n x^n = (1 + F(x))^{1/J}.$$

Then, by the stratagem of Nijenhuis and Wilf [9], we have

$$(3.20) \quad \left\{ 1 + \sum_{n=1}^{\infty} D_n x^n \right\} \sum_{k=1}^{\infty} k F_k x^{k-1} = J \sum_{n=1}^{\infty} n D_n x^{n-1} \left\{ 1 + \sum_{k=1}^{\infty} F_k x^k \right\}.$$

If we equate the coefficients of x^M on both sides of (3.20), we get for $M \geq 0$

$$(3.21) \quad (M+1)F_{M+1} + \sum_{n=1}^M D_n (M+1-n)F_{M+1-n} = J \left\{ (M+1)D_{M+1} + \sum_{n=1}^M n D_n F_{M+1-n} \right\}.$$

So for $0 \leq M$

$$(3.22) \quad D_{M+1} = \frac{1}{J(M+1)} \left\{ (M+1)F_{M+1} + \sum_{n=1}^M (M+1-n(J+1))D_n F_{M+1-n} \right\}.$$

From this, we have

$$(3.23) \quad D_1 = \frac{F_1}{J}.$$

If D_1, D_2, \dots, D_M have been calculated and stored in the computer, then by (3.22) D_{M+1} can be calculated and stored in the computer.

It will turn out eventually that we shall have calculated the first M coefficients of

$$(B(x))^{\frac{1}{2}} = B_0^{\frac{1}{2}} \left(1 + \sum_{n=1}^{\infty} D_n x^n \right)$$

by (3.22) and (3.23), and shall have calculated a number of coefficients E_1, E_2, \dots, E_M , and shall wish to solve for A_1, A_2, \dots, A_M from (3.10) and the identity

$$(3.24) \quad \sum_{n=1}^{\infty} E_n (u(z))^{2n-1} = (z B(z))^{\frac{1}{2}}.$$

By (3.10) and (3.11) we have

$$(3.25) \quad E_1 \sum_{n=1}^{\infty} A_n z^{n-1} = B_0^{\frac{1}{2}} \left(1 + \sum_{n=1}^{\infty} D_n z^n \right) - \sum_{r=2}^{\infty} E_r \left\{ \sum_{m=1}^{\infty} A_m^{(2r-1)} z^{m-2+r} \right\}.$$

Clearly

$$(3.26) \quad A_1 = \frac{B_0^{\frac{1}{2}}}{E_1}.$$

Suppose that A_1, A_2, \dots, A_N have been calculated and stored in the computer. Let us also calculate and store $A_1^{(j)}, A_2^{(j)}, \dots, A_N^{(j)}$ for $1 \leq j \leq 2N+1$, using the analogues of (3.3) and (3.4). Then by (3.25) we have

$$(3.27) \quad A_{N+1} = \frac{1}{E_1} \left\{ B_0^{\frac{1}{2}} D_N - \sum_{r=2}^{N+1} E_r A_{N+2-r}^{(2r-1)} \right\}.$$

Thus A_{N+1} can be calculated and stored.

4. A power series expansion for a Jacobi elliptic function.

We will be referring for the next few paragraphs to the NBS Handbook [3], and shall simply cite their reference numbers.

As in 17.2.1, we take m to denote the parameter. As in 17.2.18, we define the complementary parameter to be m_1 , given by

$$(4.1) \quad m_1 = 1 - m.$$

As in 17.3.1 and 17.3.5, the complete elliptic integrals are given by

$$(4.2) \quad K(m) = \int_0^{\pi/2} \frac{d\theta}{(1 - m \sin^2 \theta)^{\frac{1}{2}}}$$

$$(4.3) \quad K'(m) = K(m_1) = \int_0^{\pi/2} \frac{d\theta}{(1-m_1 \sin^2 \theta)^{1/2}} .$$

As in 17.3.17 and 17.3.18, we define the nome $q(m)$ and the complementary nome $q_1(m)$ by

$$(4.4.) \quad q(m) = \exp(-\pi K'(m)/K(m))$$

$$(4.5) \quad q_1(m) = q(m_1) = \exp(-\pi K(m)/K'(m)).$$

We observe that $K(m)$ is an increasing function of m . Hence m is a function of K , and indeed an increasing one. $K'(m)$ is a decreasing function of m , and conversely. We infer that $q(m)$ is an increasing function of m , and conversely, while $q_1(m)$ is a decreasing function of m , and conversely. As an increasing function of an increasing function is an increasing function, we infer that K is an increasing function of q . Indeed, by 17.3.22

$$(4.6) \quad \frac{2K}{\pi} = 1 + 4 \sum_{n=1}^{\infty} \frac{q^n}{1+q^{2n}} .$$

For purposes of computation, a much better series is the one given by 16.38.5

$$(4.7) \quad \left(\frac{2K}{\pi}\right)^2 = 1 + 2 \sum_{n=1}^{\infty} q^{n^2} .$$

We define

$$(4.8) \quad R(q) = \frac{2\pi}{K\sqrt{m}} .$$

By 16.38.7

$$(4.9) \quad (R(q))^{-1} = q^{\frac{1}{2}} \sum_{n=0}^{\infty} q^{n(n+1)}.$$

We define

$$(4.10) \quad S(q, v) = \sum_{n=0}^{\infty} \frac{q^{n+\frac{1}{2}}}{1-q^{2n+1}} \sin (2n+1)v.$$

Then, according to 16.23.1, the Jacobi elliptic function $\text{sn}(u|m)$ is given by

$$(4.11) \quad \text{sn}(u|m) = R(q) S(q, \frac{\pi u}{2K}).$$

We observe that

$$i S(q, -iv) = \sum_{n=0}^{\infty} \frac{q^{n+\frac{1}{2}}}{1-q^{2n+1}} \sinh (2n+1)v.$$

Clearly the right side of the equation above converges for $0 \leq v < -(\ln q)/2$. For $0 \leq v \leq -(\ln q)/2$ let us expand $\sinh (2n+1)v$ as a power series in v for each value of n . This expresses $i S(q, -iv)$ as a multiple series, each term of which is non-negative. So the multiple series is absolutely convergent. Thus, the terms can be rearranged at will. So we get

$$i S(q, -iv) = \sum_{r=0}^{\infty} (-1)^r A_r(q) v^{2r+1},$$

where we define

$$(4.12) \quad A_r(q) = \frac{(-1)^r}{(2r+1)!} \sum_{n=0}^{\infty} \frac{q^{n+\frac{1}{2}} (2n+1)^{2r+1}}{1-q^{2n+1}}.$$

As the power series shown for $iS(q, -iv)$ has all positive coefficients, and converges for $0 \leq v \leq -(\ln q)/2$, it converges absolutely for $0 \leq |v| < -(\ln q)/2$.

Thus we infer

$$(4.13) \quad S(q, v) = \sum_{r=0}^{\infty} A_r(q) v^{2r+1}.$$

Then by (4.11), we have

$$(4.14) \quad \operatorname{sn}(u|m) = R(q) \sum_{r=0}^{\infty} A_r(q) \left(\frac{\pi u}{2K} \right)^{2r+1}.$$

Professor Alan Talbot has called to our attention a much better way to calculate the coefficients of $\operatorname{sn}(u|m)$. By 17.2.2. and 17.2.7 of the N.B.S. Handbook [3], we have

$$u = \int_0^{\operatorname{sn}(u)} \{(1-t^2)(1-mt^2)\}^{-\frac{1}{2}} dt.$$

Differentiating with respect to u gives

$$(4.15) \quad \left\{ \frac{d}{du} \operatorname{sn}(u) \right\}^2 = (1 - (\operatorname{sn}(u))^2)(1 - m(\operatorname{sn}(u))^2).$$

Differentiating again gives

$$(4.16) \quad \frac{d^2}{du^2} \operatorname{sn}(u) = 2m(\operatorname{sn}(u))^3 - (1+m)\operatorname{sn}(u).$$

By (4.12) and (4.14) we have

$$(4.17) \quad \operatorname{sn}(u|m) = \sum_{r=0}^{\infty} (-1)^r g_r u^{2r+1},$$

where the g_r are positive real numbers. Taking $u = 0$ in (4.15) gives

$$(4.18) \quad g_0 = 1.$$

If we define

$$(4.19) \quad F_{2r} = 0,$$

$$(4.20) \quad F_{2r+1} = (-1)^r g_r,$$

then by (3.1) we have

$$(4.21) \quad \text{sn}(u|m) = F(u).$$

So by (3.2)

$$(4.22) \quad (\text{sn}(u|m))^3 = \sum_{n=1}^{\infty} F_n^{(3)} u^{n+2}.$$

We have

$$(4.23) \quad F_1^{(3)} = 1,$$

and by the stratagem of Nijenhuis and Wilf [9]

$$(4.24) \quad M F_{M+1}^{(3)} = \sum_{n=1}^M (3M+4-4n) F_n^{(3)} F_{M+2-n}.$$

Let us define G_r by

$$(4.25) \quad (\text{sn}(u|m))^3 = \sum_{r=0}^{\infty} (-1)^r G_r u^{2r+3}.$$

Then by (4.24)

$$(4.26) \quad (M+1) G_{M+1} = \sum_{r=0}^M (3M+3-4r) G_r g_{M+1-r}.$$

We have of course, by (4.18),

$$(4.27) \quad G_0 = 1 \quad .$$

Comparing coefficients of u in (4.16) gives

$$(4.28) \quad g_1 = \frac{1+m}{3!} \quad .$$

Substituting (4.17) and (4.25) into (4.16) gives

$$(4.29) \quad (2M+5)(2M+4)g_{M+2} = 2m G_M + (1+m)g_{M+1} \quad .$$

If we have calculated and stored g_0, g_1, \dots, g_{M+1} , then by (4.26) and (4.27) we can calculate and store G_0, G_1, \dots, G_{M+1} . Then by (4.29) we can calculate and store g_{M+2} . Hence, starting with g_0 and g_1 from (4.18) and (4.28), we can calculate g_N for as large N as desired.

We note

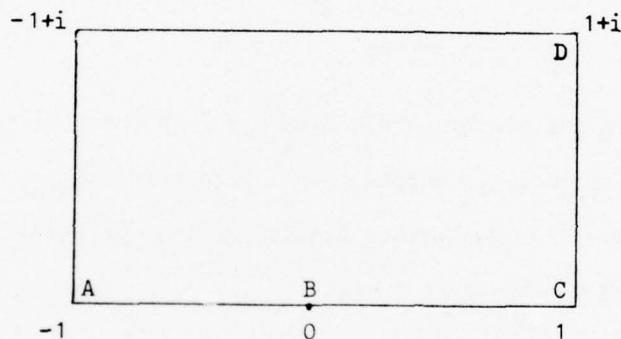
$$\begin{aligned} g_2 &= \frac{1 + 14m + m^2}{5!} \quad , \\ g_3 &= \frac{1 + 135m + 135m^2 + m^3}{7!} \quad , \\ g_4 &= \frac{1 + 1228m + 5478m^2 + 1228m^3 + m^4}{9!} \quad , \\ g_5 &= \frac{1 + 11069m + 165826m^2 + 165826m^3 + 11069m^4 + m^5}{11!} \quad , \\ G_1 &= \frac{3 + 3m}{3!} \quad , \\ G_2 &= \frac{13 + 62m + 13m^2}{5!} \quad , \\ G_3 &= \frac{205 + 3315m + 3315m^2 + 205m^3}{3(7!)} \quad . \end{aligned}$$

The values of g_0, g_1, g_2 and g_3 are given in 16.22.1 of the NBS Handbook [3], followed by the comment: "No formulae are known for the general coefficient in these series". However, given a numerical value of m , one can quickly calculate as many as might be desired of the g_r by (4.18), (4.28), (4.27), (4.26) and (4.29).

In the present report, we make a check on our calculation of the coefficients of $\text{sn}(u|m)$ by carrying out the calculation both by (4.14) and by (4.18), (4.26), (4.27), (4.28) and (4.29)..

5. A succession of conformal transformations.

We shall follow the procedure outlined in Whiteman and Papamichael [1].



The z - plane

Figure 1.

In Fig.1 we depict the rectangle of Section 1. We are using a single complex variable z , so that each point in the plane is designated by a single complex number. Thus the corners of the rectangle have the designations (± 1) and $(\pm 1+i)$. The points A, B, C, and D are located

at $-1, 0, 1$, and $1+i$. We wish an analytic function $u(z)$ whose real part has the value zero on the line segment from A to B and the value unity on the line segment from C to D, and has a zero normal derivative on the line segment from B to C and from D around to A along the top and left sides of the rectangle.

We will determine $u(z)$ by defining a succession of analytic transformations. These will entail conformal transformations of the rectangle into a succession of regions.

We first take

$$(5.1) \quad m = \frac{1}{2},$$

and write

$$(5.2) \quad z_2 \approx Kz,$$

where $K = K(m)$ is defined by (4.2). With $m = \frac{1}{2}$, we have $K'(m) = K(m)$ by (4.3). The rectangle is mapped onto a larger rectangle, with corners at $(\pm K)$ and $(\pm K + iK')$. The points A, B, C, D go into points $A_2 = -K$, $B_2 = 0$, $C_2 = K$, and $D_2 = K + iK' = K + iK$.

We now take

$$(5.3) \quad z_3 = \operatorname{sn}(z_2) = \operatorname{sn}(z_2|m).$$

The rectangle of the z_2 -plane goes into the upper half of the z_3 -plane. The points A_2, B_2, C_2 , and D_2 go into points $A_3 = -1$, $B_3 = 0$, $C_3 = 1$, and $D_3 = \sqrt{2}$.

Next we put

$$(5.4) \quad z_4 = \frac{2z_3}{1+z_3} = 2 - \frac{2}{1+z_3}.$$

The upper half of the z_3 -plane goes into the upper half of the z_4 -plane. The points A_3, B_3, C_3 , and D_3 go into points $A_4 = \infty, B_4 = 0, C_4 = 1$, and $D_4 = 2\sqrt{2}/(1+\sqrt{2})$.

Next we put

$$(5.5) \quad z_5 = \sqrt{z_4}.$$

We take the determination of the square root so that the upper half of the z_4 -plane goes into the first quadrant of the z_5 -plane. The points A_4, B_4, C_4 , and D_4 go into points $A_5 = \infty, B_5 = 0, C_5 = 1$, and $D_5 = \sqrt{4/(2+\sqrt{2})}$.

We take

$$(5.6) \quad \bar{m} = \frac{2+\sqrt{2}}{4}$$

$$(5.7) \quad z_6 = \operatorname{sn}^{-1}(z_5);$$

that is

$$(5.8) \quad z_5 = \operatorname{sn}(z_6 | \bar{m}).$$

We write

$$(5.9) \quad \bar{K} = K(\bar{m}),$$

$$(5.10) \quad \bar{K}' = K'(\bar{m}) = K(1-\bar{m});$$

see (4.2) and (4.3). Then the first quadrant of the z_5 -plane goes into a rectangle in the z_6 -plane with corners at $0, \bar{K}, i\bar{K}'$, and $\bar{K} + i\bar{K}'$. The points A_5, B_5, C_5 , and D_5 go into points $A_6 = i\bar{K}'$, $B_6 = 0$, $C_6 = \bar{K}$, and $D_6 = \bar{K} + i\bar{K}'$.

Finally, we take

$$(5.11) \quad z_7 = \frac{z_6}{\bar{K}}.$$

The rectangle of the z_6 -plane goes into a rectangle in the z_7 -plane with corners at $0, 1, i(\bar{K}'/\bar{K})$, and $1 + i(\bar{K}'/\bar{K})$. The points A_6, B_6, C_6 , and D_6 go into points $A_7 = i(\bar{K}'/\bar{K})$, $B_7 = 0$, $C_7 = 1$, and $D_7 = 1 + i(\bar{K}'/\bar{K})$.

For $r = 2, 3, \dots, 7$, we have z_r an analytic function of z . When z is on the line segment from A to B , z_7 is on the line segment from A_7 to B_7 , and has real part 0. Proceeding similarly around the rectangle of Fig. 1, we verify that z_7 is the function $u(z)$ that we were seeking. So we proceed successively to determine z_3, z_4, z_5 , and z_7 as power series in z .

6. Determination of coefficients.

By (5.2), (5.3), and (4.14), we have

$$(6.1) \quad z_3 = R(q) \sum_{r=0}^{\infty} A_r(q) \left(\frac{\pi z}{2} \right)^{2r+1}.$$

As $m = \frac{1}{2}$ by (5.1), we have $K = K'$ by (4.3). So by (4.4)

$$(6.2) \quad q = e^{-\pi} \approx 0.043213 \ 91826 \ 37722 \ 49774.$$

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

2 OF 7
ADA
046652



Then by (4.12)

$$(6.3) \quad \frac{A_0}{\sqrt{q}} \equiv 1.1847 \ 53167 \ 61544 \ 54453.$$

Similar high accuracy approximations were obtained for A_r/\sqrt{q} for $r = 1, 2, \dots, 9$.

By (4.9)

$$(6.4) \quad R(q)\sqrt{q} \equiv 0.99627 \ 55376 \ 22549 \ 34254.$$

Therefore, we had high accuracy approximations for the first 10 non-vanishing coefficients in the expansion of (6.1).

Because $m = \frac{1}{2}$ the coefficients in the expansion of $\text{sn}(u|m)$ are rational; see the discussion above commencing with formula (4.15). To cancel the powers of $\frac{1}{2}$ resulting from the powers of m , we define h_r and H_r by

$$(6.5) \quad \text{sn}(2v|\tfrac{1}{2}) = \sum_{r=0}^{\infty} (-1)^r h_r v^{2r+1},$$

$$(6.6) \quad (\text{sn}(2v|\tfrac{1}{2}))^3 = \sum_{r=0}^{\infty} (-1)^r H_r v^{2r+3}.$$

Then by (5.2) and (5.3), if we define

$$(6.7) \quad v = \frac{Kz}{2},$$

we get

$$(6.8) \quad z_3 = \sum_{r=0}^{\infty} (-1)^r h_r v^{2r+1}.$$

We record the approximation

$$(6.9) \quad \frac{K}{2} \cong 0.92703 \ 73386 \ 50685 \ 95921 \ 49 .$$

By (4.18), (4.28) and (4.27), we have

$$(6.10) \quad h_0 = 2$$

$$(6.11) \quad h_1 = 2$$

$$(6.12) \quad H_0 = 8.$$

By (4.26), we get

$$(6.13) \quad 2(M+1)H_{M+1} = \sum_{r=0}^M (3M+3-4r)H_r h_{M+1-r}$$

and by (4.29)

$$(6.14) \quad (2M+5)(2M+4)h_{M+2} = 4H_M + 6h_{M+1} .$$

Thus we get

$$(6.15) \quad z_3 = 2 \left\{ v - v^3 + \frac{11}{10} v^5 - \frac{13}{10} v^7 + \frac{181}{120} v^9 - \frac{351}{200} v^{11} \right. \\ + \frac{31861}{15600} v^{13} - \frac{1}{78000} \frac{85363}{78000} v^{15} + \frac{97}{35} \frac{77931}{36000} v^{17} \\ \left. - \frac{26}{8} \frac{25613}{16000} v^{19} + \dots \right\}$$

and

$$(6.16) \quad (z_3)^3 = 8v^3 - 24v^5 + \frac{252}{5}v^7 - 92v^9 + \frac{3851}{25}v^{11} - \frac{1217}{5}v^{13} \\ + \frac{11\,99071}{3250}v^{15} - \frac{3\,52251}{650}v^{17} + \dots$$

Using the value of $K/2$ given in (6.9), high accuracy approximations for the coefficients of z were obtained from (6.15), using (6.7). These compared to full accuracy with the approximations obtained by (6.1). This seems to corroborate that no errors were made in calculating the coefficients shown in (6.15).

From (6.15), we get by (5.4)

$$(6.17) \quad z_4 = 4v \left\{ 1 - 2v + 3v^2 - 4v^3 + \frac{51}{10}v^4 - \frac{32}{5}v^5 \right. \\ + \frac{79}{10}v^6 - \frac{48}{5}v^7 + \frac{1381}{120}v^8 - \frac{1024}{75}v^9 + \frac{3213}{200}v^{10} \\ - \frac{1408}{75}v^{11} + \frac{1\,13407}{5200}v^{12} - \frac{8192}{325}v^{13} + \frac{22\,62209}{78000}v^{14} \\ - \frac{54016}{1625}v^{15} + \frac{1342\,47851}{35\,36000}v^{16} - \frac{107\,47904}{2\,48625}v^{17} \\ \left. + \frac{1735\,68329}{35\,36000}v^{18} - \frac{230\,37952}{4\,14375}v^{19} + \dots \right\}.$$

Then by (5.5)

$$\begin{aligned}
 (6.18) \quad z_5 = 2\sqrt{v} & \left\{ 1 - v + v^2 - v^3 + \frac{21}{20} v^4 - \frac{23}{20} v^5 + \frac{25}{20} v^6 \right. \\
 & - \frac{27}{20} v^7 + \frac{3487}{2400} v^8 - \frac{1253}{800} v^9 + \frac{811}{480} v^{10} - \frac{175}{96} v^{11} \\
 & + \frac{4\,09023}{2\,08000} v^{12} - \frac{13\,23631}{6\,24000} v^{13} + \frac{2\,85557}{1\,24800} v^{14} - \frac{1\,02677}{41600} v^{15} \\
 & + \frac{13556\,81603}{5091\,84000} v^{16} - \frac{8124\,34903}{2828\,80000} v^{17} + \frac{404\,48213}{130\,56000} v^{18} \\
 & \left. - \frac{17016\,35987}{5091\,84000} v^{19} + \dots \right\}.
 \end{aligned}$$

Our ability to get rational coefficients, as above, depends on m being rational. If we had started with a rectangle of different proportions this would be quite unlikely. Of course we can always get decimal (or binary) approximations for the coefficients of z in the expansion of z_3 . This can be done either by (6.1), or by the formulae (4.18), (4.28), (4.27), (4.26) and (4.29) with a numerical approximation for each of m and K . Then, by the algorithm embodied in (3.15) and (3.17), we can get approximations for the coefficients of the powers of z in the expansion of z_4 . However, in a formula like (3.17), there is considerable danger of cancellation errors. To test this out, approximations for the coefficients of powers of z in z_4 were calculated from the approximations of powers of z in z_3 , and compared with those derived from the exact coefficients of (6.17). As expected there was cancellation error from the use of (3.17). From the coefficient of z^{11} onward, the coefficients computed by (3.17) are correct only to seventeen significant decimal accuracy, and the

coefficient of z^{19} is correct only to sixteen significant decimal accuracy. The amount of agreement is enough to corroborate that there are no errors in the coefficients shown in (6.17).

If all calculations are performed using double precision, as they are in the present report, then this amount of cancellation error is not serious. Indeed, we shall see that it is of little consequence compared with more serious errors that arise later. The fact that (3.17) can be safely used for the present rectangle gives reason to believe it can be safely used for other rectangles.

By the algorithm embodied in (3.18) through (3.23), approximations for the coefficients of the powers of z in the expansion of z_5 were calculated. There was no additional cancellation error.

As before, the errors in the calculated coefficients of z_5 are not serious if double precision is used. The agreement suffices to corroborate the coefficients shown in (6.18). Also, one feels that the algorithm will be satisfactory for rectangles of other proportions.

Recalling (6.7), let us identify (6.18) with the right side of (3.24). Recalling also that z_7 is the $u(z)$ that we seek (see the end of Section 5), we see that if we define E_n by

$$(6.19) \quad z_5 = \sum_{n=1}^{\infty} E_n z_7^{2n-1},$$

then the algorithm embodied in (3.26) and (3.27) will furnish approximations for the a_n (see (3.9) and (3.10)).

To calculate approximations for the E_n in (6.19), we proceed as follows. We define \bar{m} , \bar{K} , and \bar{K}' by (5.6), (5.9), and (5.10).

By the procedures of Rosser [4], which elaborate some of the techniques given in the NBS Handbook [3], hand calculations were performed, giving the approximations

$$(6.20) \quad \frac{\pi}{2K} \cong 0.65447 \quad 27107 \quad 79001 \quad 74587 \quad 9748$$

$$(6.21) \quad \frac{\pi}{2K'} \cong 0.96156 \quad 31078 \quad 83199 \quad 52969 \quad 2696.$$

Internal checks on the computation were as follows. We defined a_0 , b_0 , and c_0 by (2.34), (2.38), (2.40), and (2.41) of Rosser [4]. Given a_n , b_n , and c_n , we calculated a_{n+1} , b_{n+1} , and c_{n+1} by (2.39), (2.43), and (2.88) respectively of Rosser [4]. Then c_{n+1} was checked by

$$c_{n+1} = a_n - a_{n+1},$$

which follows from (2.44) and (2.39) of Rosser [4], b_n was checked by

$$b_n = a_n - 2c_{n+1},$$

which follows from (2.44) of Rosser [4], and a_{n+1} was checked by

$$(a_{n+1})^2 = (b_{n+1})^2 + (c_{n+1})^2,$$

which follows from (2.40) and (2.41) of Rosser [4]. Incidentally, the final relation furnishes at the same time a check of b_{n+1} . The approximations shown in (6.20) and (6.21) were written down from the relation

$$\frac{\pi}{2K} \cong a_5$$

and the corresponding relation for \bar{K}' (see (2.49) of Rosser [4]).

By (6.20) and (6.21), we get the approximations

$$(6.22) \quad \frac{2\bar{K}}{\pi} \cong 1.5279 \ 47588 \ 23439 \ 12569 \ 75$$

$$(6.23) \quad \pi \frac{\bar{K}'}{\bar{K}} \cong 2.1382 \ 75317 \ 86718 \ 76977 \ 62.$$

Then, by (4.4) and (4.8), with $\bar{q} = q(\bar{m})$ and $\bar{R} = R(\bar{q})$, we get

$$(6.24) \quad \bar{R} \cong 2.8335 \ 84629 \ 80564 \ 51476 \ 09$$

$$(6.25) \quad \bar{q} \cong 0.11785 \ 79353 \ 11857 \ 71914 \ 15$$

$$(6.26) \quad \bar{R} \sqrt{\bar{q}} \cong 0.97278 \ 21712 \ 72939 \ 91787.$$

As a further check, $2\bar{K}/\pi$ was recomputed from (4.7). The agreement with (6.22) gave a check of \bar{q} to 21 significant decimal digits.

Then $\bar{R} \sqrt{\bar{q}}$ was recomputed from (4.9), thereby checking (6.26) (and indirectly (6.24)), and further confirming the approximation for \bar{q} .

By (6.19), (4.14), (5.8), and (5.11), we have

$$(6.27) \quad E_{n+1} = \bar{R} \sqrt{\bar{q}} \frac{A_n(\bar{q})}{\sqrt{\bar{q}}} \left(\frac{\pi}{2} \right)^{2n+1}.$$

A double precision calculation gave the approximations

$$E_1 \cong 2.4000 \ 94459 \ 13370 \ 29880$$

$$E_2 \cong -4.2710 \ 91276 \ 71015 \ 87231$$

$$E_3 \cong 9.0780 \ 50148 \ 84322 \ 90154$$

$$E_4 \cong -19.589 \ 47213 \ 73987 \ 47225$$

$E_5 \cong$	42.271	31796	55088	65669
$E_6 \cong$	-91.247	90887	42185	21666
$E_7 \cong$	196.96	71903	29320	44722
$E_8 \cong$	-425.17	36249	74564	47812
$E_9 \cong$	917.78	04737	90827	92546
$E_{10} \cong$	-1981.1	22411	37532	06857
$E_{11} \cong$	4276.4	54055	61231	19776
$E_{12} \cong$	-9231.1	60670	91645	55305
$E_{13} \cong$	19926.	39841	97583	22048
$E_{14} \cong$	-43013.	15599	83815	48733
$E_{15} \cong$	92848.	26841	09464	34459
$E_{16} \cong$	-20042	2.4230	19509	75164
$E_{17} \cong$	43263	2.1679	06704	27497
$E_{18} \cong$	-93388	0.5004	34001	30625
$E_{19} \cong$	20158	75.965	28174	33942
$E_{20} \cong$	-43514	73.133	35277	44512

Approximations of the coefficients E_n were also calculated, using double precision, by the algorithm embodied in (4.15) through (4.29). There was agreement between the two sets of approximations to seventeen significant decimal accuracy.

In preparation for the use of (3.26) and (3.27) we substitute (6.7)

into (6.18) to get

$$(6.28) \quad z_5 = \sqrt{z} (B_0^{\frac{1}{2}} + \sum_{n=1}^{\infty} (B_0^{\frac{1}{2}} D_n) z^n) .$$

From the succession of transformations in Section 5, it is clear that z_5 is infinite when $z = -1$. Indeed, z_5 has a pole at $z = -1$, so that

$$(-1)^n B_0^{\frac{1}{2}} D_n$$

should approach a limit as n goes to infinity. From our calculated approximations, it appears that this limit is approximately

$$1.52552.$$

So, by (3.27), we see that A_{N+1} is calculated by subtracting a sum from a number approximately equal to 1.5 and then dividing by E_1 , which is approximately 2.4. As $|A_{N+1}|$ is quite small for the larger values of N , there will have to be quite severe cancellation errors. In addition, there will be rapid growth of round off errors as we progress to larger N . In (3.27) the coefficient of A_N on the right is

$$-3 E_2 A_1^2 / E_1 .$$

As we shall see shortly,

$$A_1 \approx 0.80232 .$$

Thus, to compute A_{N+1} , we multiply A_N by approximately 3.4. Thus we multiply the error of A_N by a similar amount. So it appears

that in going from A_N to A_{N+2} we can expect to lose about one decimal digit of accuracy on the right due to round off error. That is, we cannot expect A_{20} to have more than 10 decimal digits correct to the right of the decimal point.

With this in mind, we write down the approximations we obtained for the A_n . We have dropped off the final digits that are almost certainly wrong. Perhaps the last digit or two that we list is in error, but we rather doubt if any more are.

$$A_1 \cong 0.80232 \ 49074 \ 90468 \ 84047$$

$$A_2 \cong 0.17531 \ 18403 \ 90175 \ 83405$$

$$A_3 \cong 0.03447 \ 58301 \ 58893 \ 6187$$

$$A_4 \cong -0.01614 \ 24305 \ 19396 \ 2687$$

$$A_5 \cong 0.00288 \ 05454 \ 34045 \ 715$$

$$A_6 \cong 0.00066 \ 21097 \ 71841 \ 473$$

$$A_7 \cong 0.00055 \ 08746 \ 89018 \ 36$$

$$A_8 \cong -0.00017 \ 38659 \ 89051 \ 07$$

$$A_9 \cong 0.00006 \ 72097 \ 56853 \ 1$$

$$A_{10} \cong 0.00003 \ 07687 \ 48965 \ 1$$

$$A_{11} \cong 0.00001 \ 46046 \ 03348$$

$$A_{12} \cong -0.00000 \ 63682 \ 27833$$

$$A_{13} \cong 0.00000 \ 24412 \ 9222$$

$$A_{14} \cong 0.00000 \ 10619 \ 3096$$

$$A_{15} \approx 0.00000 \quad 05430 \quad 244$$

$$A_{16} \approx -0.00000 \quad 02400 \quad 927$$

$$A_{17} \approx 0.00000 \quad 01010 \quad 80$$

$$A_{18} \approx 0.00000 \quad 00463 \quad 34$$

$$A_{19} \approx 0.00000 \quad 00230 \quad 7$$

$$A_{20} \approx -0.00000 \quad 00105 \quad 9.$$

If z_7 is bounded for $|z| = 2$, as it most likely is, then by the Cauchy integral theorem there is a constant k such that for $1 \leq n$

$$|A_n| \leq \frac{k}{2^n}.$$

At first, the A_n drop off much more rapidly than this, but it appears from the tabulated approximations that, from about A_8 onwards, the value

$$\left| \frac{A_n}{A_{n+1}} \right|$$

does not change greatly with n , and does appear to be tending towards 2. This hints at a tantalizingly uniform behavior of the A_n . In this connection one is struck by the fact that, as far as our tabulation goes, every fourth A_n is negative.

For $z = 1$, the sum of the series should be unity. The sum of the coefficients shown is

$$0.99999 \quad 99924.$$

If, in accordance with the "uniformity" we observed above, we guess

$$A_{21} \approx 50 \times 10^{-10}$$

$$A_{22} \approx 20 \times 10^{-10}$$

$$A_{23} \approx 8 \times 10^{-10}$$

$$A_{24} \approx -3 \times 10^{-10}$$

$$A_{25} \approx 1 \times 10^{-10},$$

we will get the desired sum of unity.

In Whiteman and Papamichael [1], in Tables 1 and 2, are listed values of $R(z_7)$ for a variety of values of z . The series was summed for these values of z , and the real part was compared with the eight place values from which the entries in Tables 1 and 2 of Whiteman and Papamichael [1] were rounded. The agreement was perfect, except for values of z with $|z| \geq 1$, for which the absence of the coefficients A_{21}, A_{22}, \dots led to minor discrepancies.

7. Other boundary conditions.

In Whiteman and Papamichael [1], a conformal map from the rectangle in Fig.1 of Section 5 to the final rectangle in the z_7 -plane (see Section 5) was derived, essentially as we have explained it in Section 5. Our coefficients A_n are merely the coefficients of the power series expansion of this transformation. Given a point in the z -plane, one can find the corresponding point in the z_7 -plane either by the methods of Whiteman and Papamichael [1], or by using our approximations for the coefficients to compute an approximate sum for the series. We have given enough coefficients that one can get high accuracy except

near the two upper corners. It would be a straightforward calculation to get more coefficients, so that one can get high accuracy everywhere in the rectangle. One would have to use higher precision than double precision, but most large computing centers have software capable of doing this.

The point of the conformal transformation is that boundary conditions which are difficult to satisfy in the rectangle in Fig.1 may be transformed into boundary conditions which are easy to satisfy in the rectangle in the z_7 -plane. Thus the difficult boundary conditions (1.2) through (1.6) are transformed into entirely trivial boundary conditions for the rectangle in the z_7 -plane.

If one should modify the Dirichlet conditions (1.2) and (1.6), one would have in the z_7 -plane Dirichlet conditions on two ends, and zero normal derivatives on the other two sides. Such problems are fairly routine. If one should modify the condition (1.3) to

$$(7.1) \quad u_y(x,0) = f(x) \quad 0 < x < 1,$$

one would still have for the rectangle in the z_7 -plane Dirichlet conditions on two ends and Neumann conditions on the other two sides. There are familiar techniques which are quite adequate to deal with this, except for one difficulty. That is to determine what is the Neumann condition in the z_7 -plane corresponding to (7.1). We now have a way to deal with this.

As we noted in Section 1, $u(x,y)$ is the real part of an analytic function $u(z)$. But, by our transformations, z is an analytic function of z_7 . Thus, if we define

$$(7.2) \quad u_7(z_7) = u(z) ,$$

then $u_7(z_7)$ is an analytic function of z_7 . Let us put

$$(7.3) \quad z_7 = \xi + i \eta .$$

Then the real part of u_7 will be a harmonic function of ξ and η . We denote this by

$$(7.4) \quad u_7(\xi, \eta) .$$

We wish to find the condition corresponding to (7.1).

As z_7 is a function of z , we have by (1.7) and (7.3) that ξ and η are functions of x and y . Indeed, we have

$$(7.5) \quad u_7(\xi, \eta) = u(x, y) .$$

So

$$(7.6) \quad \frac{\partial}{\partial y} u(x, y) = \frac{\partial}{\partial \xi} u_7(\xi, \eta) \frac{\partial \xi}{\partial y} + \frac{\partial}{\partial \eta} u_7(\xi, \eta) \frac{\partial \eta}{\partial y} .$$

For $0 < x < 1$ and $y = 0$, we have $0 < \xi < 1$ and $\eta = 0$.

So

$$\frac{\partial \eta}{\partial x} = 0 .$$

By the Cauchy-Riemann differential equations, this gives

$$\frac{\partial \xi}{\partial y} = 0.$$

Then, by the Cauchy-Riemann differential equations, (7.6) reduces to

$$(7.7) \quad \frac{\partial}{\partial y} u(x, 0) = \frac{\partial}{\partial \eta} u_7(\xi, 0) \frac{\partial \xi}{\partial x}.$$

We have

$$(7.8) \quad z_7 = \sum_{n=1}^{\infty} A_n z^{n-\frac{1}{2}}.$$

But the A_n are all real, so that for $y = 0$ and $0 < x < 1$ we have

$$(7.9) \quad \frac{\partial \xi}{\partial x} = \sum_{n=1}^{\infty} (n - \frac{1}{2}) A_n x^{n-(3/2)}.$$

Then, by (7.7) and (7.1), we can determine the values of $\partial u_7(\xi, \eta) / \partial \eta$ for $\eta = 0$.

The series on the right of (7.9) is more slowly converging than the one on the right of (7.8). However, in (7.1) we have $0 < x < 1$, and one can probably get adequate accuracy by means of the coefficients which we have given.

Along the left side and top of the rectangle in Fig. 1, the treatment we have given for (7.6) and (7.7) would be replaced by something more complex. Also, we would replace (7.9) by a more involved relation. Worse than that, one would get into parts of the boundary where the convergence would be fairly slow, so that one

would have to determine more coefficients.

As noted above, more coefficients can be determined, but it is laborious. So we seek a better procedure. What we shall do is to specify an auxiliary function $u^*(x,y)$ which satisfies the desired Neumann conditions along the left side and top of the rectangle in Fig.1. Also $u^*(x,y)$ and its partial derivatives can be calculated at points (x,y) inside or on the boundary of the rectangle. Then $u(x,y) - u^*(x,y)$ has zero normal derivatives along the left side and top of the rectangle. Also, we can determine the conditions for $u(x,y) - u^*(x,y)$ and its normal derivatives along the rest of the boundary. Thus, we can determine $u(x,y) - u^*(x,y)$ by the method indicated earlier in the section. Then $u(x,y)$ can be determined, since we can calculate $u^*(x,y)$.

By rotating and translating the rectangle in Fig.1, we can restrict attention to the case in which the rectangle lies in the first quadrant, with its lower left hand corner at the origin, and we wish to choose $u^*(x,y)$ to satisfy stated Neumann conditions along the x-axis and y-axis. Nothing is specified as to what $u^*(x,y)$ shall do on the other two sides of the rectangle. More than that, it suffices to consider the special case in which the normal derivative is zero along the imaginary axis; by reflecting about the 45° line $y = x$, we can interchange the x-axis and y-axis.

So let the rectangle have parts of the x-axis and y-axis as its bottom and left side, and let it extend from 0 to A along the x-axis. Let us find $u^*(x,y)$ which is harmonic inside the rectangle, such that

$$(7.10) \quad u_x^*(0, y) = 0 \quad 0 < y$$

$$(7.11) \quad u_y^*(x, 0) = g(x) \quad 0 < x < A.$$

We shall assume that $g(x)$ is of bounded variation for $0 \leq x \leq A$.

We shall show that

$$(7.12) \quad u^*(x, y) = \frac{1}{2\pi} \int_0^A g(t) \log \{(x-t)^2 + y^2\} dt. \\ + \frac{1}{2\pi} \int_0^A g(t) \log \{(x+t)^2 + y^2\} dt$$

fulfills the stated requirements. Since $g(t)$ is of bounded variation, we can form partial derivatives by differentiating under the integral sign. So it is easily seen that $u^*(x, y)$ is harmonic.

Also

$$u_x^*(x, y) = \frac{1}{\pi} \int_0^A \frac{g(t)(x-t)dt}{(x-t)^2 + y^2} + \frac{1}{\pi} \int_0^A \frac{g(t)(x+t)dt}{(x+t)^2 + y^2}.$$

So (7.10) is satisfied.

We have

$$(7.13) \quad u_y^*(x, y) = \frac{1}{\pi} \int_0^A \frac{g(t)y dt}{(x-t)^2 + y^2} + \frac{1}{\pi} \int_0^A \frac{g(t)y dt}{(x+t)^2 + y^2}.$$

Clearly the second integral on the right approaches zero as $y \rightarrow 0$.

As $g(t)$ is of bounded variation, it has limits $g(x-0)$ and $g(x+0)$ as t approaches x from the left and from the right

respectively. We write

$$(7.14) \quad \frac{1}{\pi} \int_0^A \frac{g(t)y \, dt}{(x-t)^2 + y^2} = \frac{1}{\pi} \int_0^x \frac{\{g(t)-g(x-0)\}y \, dt}{(x-t)^2 + y^2} + \frac{1}{\pi} \int_x^A \frac{\{g(t)-g(x+0)\}y \, dt}{(x-t)^2 + y^2} \\ + \frac{1}{\pi} \int_0^x \frac{g(x-0)y \, dt}{(x-t)^2 + y^2} + \frac{1}{\pi} \int_x^A \frac{g(x+0)y \, dt}{(x-t)^2 + y^2} .$$

We note that

$$\int_0^x \frac{y \, dt}{(x-t)^2 + y^2} = - \arctan \frac{x-t}{y} \Big|_0^x = \arctan \frac{x}{y} , \\ \int_x^A \frac{y \, dt}{(x-t)^2 + y^2} = \arctan \frac{t-x}{y} \Big|_x^A = \arctan \frac{A-x}{y} .$$

So, for $0 < x < A$,

$$(7.15) \quad \lim_{y \rightarrow 0} \left\{ \frac{1}{\pi} \int_0^x \frac{g(x-0)y \, dt}{(x-t)^2 + y^2} + \frac{1}{\pi} \int_x^A \frac{g(x+0)y \, dt}{(x-t)^2 + y^2} \right\} = \frac{1}{2} \{g(x-0) + g(x+0)\} .$$

We will now show that

$$(7.16) \quad \lim_{y \rightarrow 0} \frac{1}{\pi} \int_0^x \frac{\{g(t)-g(x-0)\}y \, dt}{(x-t)^2 + y^2} = 0 .$$

Specifically, given any $\delta > 0$, we will show that for all y sufficiently close to 0, the quantity in question is less than δ in absolute value.

Given $\delta > 0$ choose $\epsilon > 0$ sufficiently small so that

$$|g(t) - g(x-0)| < \delta$$

for $x - \epsilon \leq t < x$. Then for $0 < y$

$$\left| \frac{1}{\pi} \int_{x-\epsilon}^x \frac{\{g(t)-g(x-0)\}y \, dt}{(x-t)^2 + y^2} \right| < \frac{\delta}{\pi} \int_{x-\epsilon}^x \frac{y \, dt}{(x-t)^2 + y^2} = \frac{\delta}{\pi} \arctan \frac{\epsilon}{y} < \frac{\delta}{2}.$$

Choose M so that

$$|g(x-0)| < M$$

for $0 \leq t \leq x$ and for $0 < y$

$$\left| \frac{1}{\pi} \int_0^{x-\epsilon} \frac{\{g(t)-g(x-0)\}y \, dt}{(x-t)^2 + y^2} \right| < \frac{My}{\pi} \int_0^{x-\epsilon} \frac{dt}{(x-t)^2 + y^2} < \frac{My}{\pi} \int_0^{x-\epsilon} \frac{dt}{(x-t)^2} < \frac{My}{\pi\epsilon}.$$

For $0 < y < (\pi\delta\epsilon)/(2M)$, we have

$$\left| \frac{1}{\pi} \int_0^{x-\epsilon} \frac{\{g(t)-g(x-0)\}y \, dt}{(x-t)^2 + y^2} \right| < \frac{\delta}{2}.$$

In the same way, we show

$$(7.17) \quad \lim_{y \rightarrow 0} \frac{1}{\pi} \int_x^A \frac{\{g(t)-g(x+0)\}y \, dt}{(x-t)^2 + y^2} = 0.$$

So, by (7.13), (7.14), (7.15), (7.16), and (7.17), we conclude that

$$\lim_{y \rightarrow 0} u_y^*(x, y) = \frac{1}{2} \{g(x-0) + g(x+0)\}.$$

If $g(t)$ is continuous at $t = x$, then (7.11) holds.

If $g(t)$ is discontinuous at $t = x_0$, then $u_y^*(x,y)$ can be made to approach any limit between $g(x_0 - 0)$ and $g(x_0 + 0)$ by approaching the point $(x_0, 0)$ along a suitable direction. However, this would have to be the case for any $u^*(x,y)$ which satisfies (7.11). Thus our $u^*(x,y)$ comes as close to satisfying (7.11) at $x = x_0$ as is possible.

For a given value of x and y , one can approximate the $u^*(x,y)$ of (7.12) by numerical quadrature. If $y = 0$, one could avoid singularities by putting

$$t = x \pm e^{-s}.$$

Thus

$$\begin{aligned} \frac{1}{2\pi} \int_0^A g(t) \log(x-t)^2 dt &= -\frac{1}{\pi} \int_{-\log x}^{\infty} g(x-e^{-s}) s e^{-s} ds \\ &- \frac{1}{\pi} \int_{-\log(A-x)}^{\infty} g(x+e^{-s}) s e^{-s} ds. \end{aligned}$$

For this, a numerical quadrature would be quite satisfactory; one would carry the integration out to some large value of s , rather than to infinity, of course.

If one should wish values of $u^*(x,y)$ at a great many points, a reasonable procedure would be to approximate values of $u^*(x,y)$ on a grid by numerical quadrature, and then to interpolate by the spline function interpolation method of Papamichael and Whiteman [5].

For either of the integrals appearing on the right of (7.12),

numerical quadrature will be quite unsatisfactory near $x = y = 0$. However, the irregularities in the two integrals will mostly cancel out if we combine the two integrals into a single integral. Define

$$(7.18) \quad g^{**}(t) = \begin{cases} g(t) & 0 \leq t < A \\ g(-t) & -A < t \leq 0 \end{cases}$$

and then (7.12) takes the form

$$(7.19) \quad u^*(x, y) = \frac{1}{2\pi} \int_{-A}^A g^{**}(t) \log\{(x-t)^2 + y^2\} dt.$$

In this, numerical quadrature is still unsatisfactory near $x = A$, $y = 0$. To improve this, define $g^*(t)$ to satisfy

$$(7.20) \quad g^*(t) = g^*(-t)$$

$$(7.21) \quad g^*(t) = g^{**}(t) \quad -A < t < A$$

with $g^*(t)$ as smooth as possible for $A \leq t \leq B$, where $B > A$; if $g(t)$ has a left derivative at $t = A$, we can (and should) choose $g^*(t)$ to have a continuous derivative for $A \leq t \leq B$. Then in place of (7.12), we can define

$$(7.22) \quad u^*(x, y) = \frac{1}{2\pi} \int_{-B}^B g^*(t) \log\{(x-t)^2 + y^2\} dt.$$

This will be a slightly different function from that defined by (7.12), but we can use the same argument as we gave for (7.12)

to show that

$$u_x^*(0,y) = 0 \quad 0 < y$$

and

$$\lim_{y \rightarrow 0} u_y^*(x,y) = \frac{1}{2} \{g(x-0) + g(x+0)\}$$

holds for $0 < x < A$; at $x = 0$ and $x = A$ the limits will be $g(0+)$ and $g(A-0)$ respectively.

If $g^*(t)$ is discontinuous at $t = b$, with $0 < b < A$, then numerical quadrature of the right side of (7.22) will be very poor near $x = b$, $y = 0$. It would likely be worthwhile to use the techniques of Section 7 of Rosser [6] to "remove" discontinuities of $g^*(t)$ before defining $u^*(x,y)$. Somewhat the same is true of discontinuities of the derivatives of $g^*(t)$. Thus, if $g(t)$ has a right-hand derivative at $t = 0$ which is not 0, then the derivative of $g^*(t)$ will have a discontinuity at $t = 0$ which can be "removed" by the techniques of Section 7 of Rosser [6]. This "removal" would improve the numerical quadrature of the right side of (7.22). Certainly, the smoother $g^*(t)$ is, the better will be the numerical quadrature of the right side of (7.22).

REFERENCES

- [1] J.R.Whiteman and N.Papamichael, "Treatment of harmonic mixed boundary problems by conformal transformation methods," Journal of Applied Mathematics and Physics (ZAMP), vol.23 (1972), pp.655-664.
- [2] Wolfgang Wasow, "Asymptotic development of the solution of Dirichlet's problem at analytic corners," Duke Mathematical Journal, vol.24 (1957), pp 47-56.
- [3] Milton Abramowitz and Irene Stegun, editors, "Handbook of mathematical functions," National Bureau of Standards Applied Mathematics Series, No.55, U.S.Government Printing Office, 1964.
- [4] J.Barkley Rosser, "Potentials of charged plates, Part I, Mathematics Research Center Technical Summary Report # 1318, January 15, 1973.
- [5] N.Papamichael and John R.Whiteman, "Cubic spline interpolation of harmonic functions," Technical Report TR/28, Dept.of Mathematics, Brunel University, 1973.
- [6] J.Barkley Rosser, "Finite-difference solution of Poisson's equation in rectangles of arbitrary shape," Technical Report TR/27, Dept.of Mathematics, Brunel University, 1973, and MRC Technical Summary Report # 1404, 1973.
- [7] M.Tsuji, "Potential theory in modern function theory" Maruzen Co.Ltd., Tokyo, 1959.
- [8] O.D.Kellogg, "Foundations of potential theory" Dover Publications, Inc., New York, 1953.
- [9] A.Nijenhuis and H.S.Wilf, "Combinatorial algorithms," Academic Press 1975.

A FOURIER-SOLUTION OF PARABOLIC PDE BY TAYLOR SERIES

Y. F. Chang
Computer Science Department
University of Nebraska,
Lincoln, Nebraska 68588

ABSTRACT. Two-dimensional Taylor series are used to solve initial-boundary value problems in parabolic partial differential equations. Earlier works by the author dealt with basic concepts in the solutions of parabolic PDE's by Taylor series. The initial and boundary conditions were then restricted to be compatible (i.e. either initial or boundary conditions alone are sufficient to solve the problem). This paper will deal with problems where the initial and boundary conditions are incompatible and even inconsistent. The solution is obtained using two-dimensional Taylor series in a manner analogous to Fourier-series solutions. The following example is solved to illustrate this method of Taylor series.

$$u_{xx} - x^2 \cdot u = u_t, \text{ satisfying } u(0,t) = u(2,t) = 0, \text{ and } u(x,0) = 1.$$

The solution proceeds as follows. Using the differential equation as the recursive relation, a family of two-dimensional Taylor series is constructed defining functions which satisfy both the equation and the boundary conditions, and which resemble the functions $\sin(n\pi x/2)$. The final solution is a linear combination of the family of two-dimensional series with coefficients determined by a Fourier-type approximation of the initial condition.

1. INTRODUCTION. The application of Taylor series in the solution of ordinary differential equations has been treated with growing interest in the published literature. For example, study the works of Gibbons(1), Richtmeyer(2), Fehlberg(3), Hartwell(4), Leavitt(5), Barton, et al(6), and Chang(7). At present, there is a need to explore the application of multi-dimensional Taylor series to the solutions of partial differential equations (PDE).

The solution of parabolic PDE's with compatible initial and boundary conditions has been discussed by Chang(8) & (9). A compatible problem is one for which the initial condition (or the boundary conditions) alone is sufficient to uniquely determine the solution to the problem. When the boundary conditions are used to determine the solution, as in Chang(8), it is necessary to use three-dimensional Taylor series, whose terms are constructed recursively from the PDE and the boundary conditions. This is accomplished without iteration. When the initial condition is used to determine the solution, as in Chang(9), it is necessary to use two-dimensional Taylor series, whose terms are constructed recursively from the PDE and the initial condition. This solution also is accomplished without iteration.

In this paper, we will discuss the solution of parabolic PDE with incompatible and even inconsistent initial and boundary conditions. The following sample problem will be solved to illustrate this method of two-dimensional Taylor series solution.

$$u_{xx} - x^2 \cdot u = u_t, \text{ satisfying } u(0,t) = u(2,t) = 0, \text{ and } u(x,0) = 1. \quad (1)$$

We will first briefly review some elementary concepts in the Fourier solution of the heat equation.

$$u_{xx} = u_t, \text{ satisfying } u(0,t) = u(2,t) = 0, \text{ and } u(x,0) = 1. \quad (2)$$

Consider the family of functions $f_n(x) = \sin(n\pi x/2)$, with $n=1,2,3,\dots$, which satisfy the boundary conditions $f_n(0) = f_n(2) = 0$ for all n . Next, consider the functions

$$y_n(x,t) = \exp(-a^2 t) \cdot f_n(x). \quad (3)$$

The y -functions satisfy the heat equation if $a = n\pi x/2$. The solution of Eq.(2) is found as the linear combination of the y -functions,

$$u(x,t) = \sum_{n=1}^{\infty} c_n \cdot y_n(x,t),$$

where the coefficients c_n are determined from the initial condition. Before determining c_n , it is best to orthonormalize $y_n(x,t)$ in the interval $[0,2]$. Then, the coefficients are found by the inner product

$$c_n = \int_0^2 y_n(x,0) \cdot u(x,0) dx = \int_0^2 f_n(x) \cdot u(x,0) dx,$$

where $u(x,0)$ can be any given initial condition.

2. DISCUSSION OF TAYLOR SERIES METHOD. The development of the Taylor series method for the solution of parabolic PDE's begins with a detailed examination of the solution functions $y_n(x,t)$, see Eq.(3). We expand $y_n(x,t)$ in a two-dimensional Taylor series about the point $x=0$ and $t=0$, with an increment h in x and an increment g in t . The result is shown in Table I, with orders of x placed horizontally and orders of t placed vertically.

There are four important features to be noted concerning Table I. (a) Every odd column in Table I contains only zero elements, so $y_n(x,t)$ satisfies the boundary condition $u(0,t) = 0$. (b) The sum of all the terms in each individual row is zero for $a = n\pi x/2$ and $h = 2$, so $y_n(x,t)$ satisfies the boundary condition $u(2,t) = 0$. (c) There are pairs of terms in this 2-D array that satisfy the recursive relation

$$Y(n,m) = \frac{h^2}{g} \frac{m}{(n-1)(n-2)} Y(n-2,m+1), \quad (4)$$

where n and m are the orders of the terms in the x and t directions. This recursive relation is derivable from the heat equation. (d) The adjacent rows differ by a constant. For example, rows #1 and #2 differ by the constant $-a^2 g$ and rows #3 and #4 differ by the constant $-a^2 g/3$. This is an important observation that will aid in the solution of the more complex problem.

Table I. The Two-Dimensional Array for $y_n(x,t)$.

$\begin{smallmatrix} x \\ t \end{smallmatrix}$	1	2	3	4	5	6	7	8
1	0	ah	0	$\frac{-a^3 h^3}{3!}$	0	$\frac{a^5 h^5}{5!}$	0	$\frac{-a^7 h^7}{7!}$
2	0	$-a^3 hg$	0	$\frac{a^5 h^3 g}{3!}$	0	$\frac{-a^7 h^5 g}{5!}$	0	$\frac{a^9 h^7 g}{7!}$
3	0	$\frac{a^5 h g^2}{2!}$	0	$\frac{-a^7 h^3 g^2}{2! 3!}$	0	$\frac{a^9 h^5 g^2}{2! 5!}$	0	$\frac{-a^{11} h^7 g^2}{2! 7!}$
4	0	$\frac{-a^7 h g^3}{3!}$	0	$\frac{a^9 h^3 g^3}{3! 3!}$	0	$\frac{-a^{11} h^5 g^3}{3! 5!}$	0	$\frac{a^{13} h^7 g^3}{3! 7!}$
5	0	$\frac{a^9 h g^4}{4!}$	0	$\frac{-a^{11} h^3 g^4}{4! 3!}$	0	$\frac{a^{13} h^5 g^4}{4! 5!}$	0	$\frac{-a^{15} h^7 g^4}{4! 7!}$

We will derive Eq.(4) from the heat equation and extend that to the problem in Eq.(1). Consider a reduced derivative defined as

$$Y(n+1, m+1) = \frac{\partial^{n+m} y}{\partial x^n \partial t^m} \frac{h^n}{n!} \frac{g^m}{m!}, \quad n, m = 0, 1, 2, \dots \quad (5)$$

where y is a function of x and t , h is the increment in x , and g is the increment in t . (The orders $n+1$ and $m+1$ are necessary due to a limitation of FORTRAN.) In this paper, upper-case letters are reserved for the reduced derivatives of the underlying lower-case functions. The reduced derivatives are the terms of the two-dimensional Taylor series calculated at a specific point $(0,0)$ with increments h and g . Given a function $f(x,t)$, the value of f at $(0,0)$ is stored as $F(1,1)$, the value of $\partial^n f / \partial x^n$ at $(0,0)$ is stored in $F(n+1,1)$, and the value of $\partial^m f / \partial t^m$ at $(0,0)$ is stored in $F(1,m+1)$. Graphically, given a 2-D array for $f(x,t)$, differentiation of f with respect to x n -times yields the term n -spaces to the right, and differentiation of f with respect to t m -times yields the term m -spaces down.

Consider the function $w(x,t) = \partial^2 u / \partial x^2$, or in terms of reduced derivatives $W(1,1) = 2! \cdot U(3,1) / h^2$. Differentiation of w with respect to t m -times yields the term m -spaces down from $U(3,1)$ in the 2-D U -array. The result is $W(1,m+1) = 2! \cdot U(3,m+1) / h^2$. However, differentiation of w with respect to x does not yield the term one space to the right of $U(3,1)$ in the U -array. The term $W(2,1)$ is not equal to $2! \cdot U(4,1) / h^2$. This is due to a mismatch in the factorial functions between the W -array and the U -array. Observe that the true relations are

$$W(n,1) = \frac{n(n+1)}{h^2} U(n+2,1), \text{ and } W(n,m) = \frac{n(n+1)}{h^2} U(n+2,m). \quad (6)$$

Therefore, differentiation of w with respect to x n -times yields the term n -spaces to the right of $U(3,1)$ multiplied by $n(n+1)/h^2$. The same analysis applies for the function $v(x,t) = \partial u / \partial t$, with the result

$$V(n,m) = \frac{m}{g} U(n,m+1) . \quad (7)$$

Consider next Leibnitz' Rule for the derivatives of a product $y(x,t) = p(x,t) \cdot q(x,t)$

$$\frac{1}{n!} \frac{\partial^n y}{\partial x^n} = \sum_{i=0}^n \frac{1}{i!} \frac{\partial^i p}{\partial x^i} \frac{1}{(n-i)!} \frac{\partial^{n-i} q}{\partial x^{n-i}} .$$

In terms of reduced derivatives, the modified Leibnitz' Rule is

$$Y(n,1) = \sum_{i=1}^n P(i,1) \cdot Q(n-i+1,1) . \quad (8)$$

The modified Leibnitz' Rule will be very useful in the analysis of the PDE in Eq.(1).

We will now derive the recursive relation, see Eq.(4), from the heat equation using two different approaches. Differentiation of the heat equation with respect to x n -times and with respect to t m -times yields the general form

$$\frac{\partial^{n+m+2} u}{\partial x^{n+2} \partial t^m} = \frac{\partial^{n+m+1} u}{\partial x^n \partial t^{m+1}} ,$$

which when written in terms of the reduced derivatives becomes

$$\frac{(n+1)(n+2)}{h^2} U(n+3,m+1) = \frac{m+1}{g} U(n+1,m+2) .$$

With adjustment for the orders, this is the same recursive relation as that given in Eq.(4).

The second approach begins with the heat equation written in terms of the reduced derivatives

$$\frac{1 \cdot 2}{h^2} U(3,1) = \frac{1}{g} U(1,2) . \quad (9)$$

Recognizing the relations in Eq.(6) and in Eq.(7), the repeated differentiation of the functions represented by the terms in Eq.(9) with respect to x and t yields the recursive relation

$$\frac{n(n+1)}{h^2} U(n+2,m) = \frac{m}{g} U(n,m+1) .$$

Except for the orders, this recursive relation is the same as those above.

Next, we will derive the recursive relation for the parabolic PDE of Eq.(1)

$$u_{xx} = x^2 \cdot u + u_t \quad (10)$$

Repeated differentiation of Eq.(10) with respect to x n -times yields

$$\frac{\partial^{n+2} u}{\partial x^{n+2}} = x^2 \frac{\partial^n u}{\partial x^n} + 2nx \frac{\partial^{n-1} u}{\partial x^{n-1}} + n(n-1) \frac{\partial^{n-2} u}{\partial x^{n-2}} + \frac{\partial^{n+1} u}{\partial x^n \partial t} \quad (11)$$

Just as in the case of the heat equation, we will expand the solution function of Eq.(10) about $x=0$ and $t=0$. With this condition, Eq.(11) becomes

$$\frac{\partial^{n+2} u}{\partial x^{n+2}} = n(n-1) \frac{\partial^{n-2} u}{\partial x^{n-2}} + \frac{\partial^{n+1} u}{\partial x^n \partial t} \quad (12)$$

Repeated differentiation of Eq.(12) with respect to t m -times yields the general form

$$\frac{\partial^{n+m+2} u}{\partial x^{n+2} \partial t^m} = n(n-1) \frac{\partial^{n+m-2} u}{\partial x^{n-2} \partial t^m} + \frac{\partial^{n+m+1} u}{\partial x^n \partial t^{m+1}} \quad (13)$$

When written in terms of the reduced derivatives, Eq.(13) becomes the desired recursive relation

$$\frac{(n+1)(n+2)}{h^2} U(n+3, m+1) = h^2 U(n-1, m+1) + \frac{m+1}{g} U(n+1, m+2) \quad (14)$$

We will derive Eq.(14) from a second approach. Written in terms of the reduced derivatives, Eq.(10) becomes

$$\frac{1 \cdot 2}{h^2} U(3, 1) = x^2 U(1, 1) + \frac{1}{g} U(1, 2) \quad (15)$$

Applying the modified Leibnitz' Rule to the product $f = x^2 \cdot u$, we find

$$F(n, 1) = x^2 U(n, 1) + 2xh \cdot U(n-1, 1) + h^2 \cdot U(n-2, 1) \quad (16)$$

With $x=0$, this becomes

$$F(n, 1) = h^2 U(n-2, 1) \quad (16)$$

Recognizing the relations in Eq.(6) and in Eq.(7), and applying the result in Eq.(16), the repeated differentiation of the functions represented by the terms in Eq.(15) with respect to x n -times and t m -times yields the result

$$\frac{n(n+1)}{h^2} U(n+2, m) = h^2 U(n-2, m) + \frac{m}{g} U(n, m+1) \quad (17)$$

Except for adjustment of orders, Eq.(14) and Eq.(17) are identical. This is

the recursive relation that we will use to derive the 2-D array solution function for Eq.(10).

The boundary condition at $x=0$ is satisfied if the first column in the 2-D array has only zero elements. A simple examination of Eq.(17) shows that the recursive relation is between terms in alternate columns. Therefore, since the first column contains only zero elements, all odd columns in the array will contain only zero elements. This result is the same as that for the heat equation.

In the 2-D array for the solution functions of the heat equation, the adjacent rows differ by a constant. We will show that the same is true for the solution functions of Eq.(10). Let the second-column terms be given by

$$Y(2,1) = a \cdot h \quad , \quad Y(2,2) = k_2 a \cdot h \cdot g \quad ,$$

$$\text{and} \quad Y(2,m) = k_m a \cdot h \cdot g^{m-1} / (m-1)! \quad , \quad (18)$$

where $k_m g^{m-1} / (m-1)!$ is the constant multiplier between the higher-order rows and the first row. By Eq.(17), the fourth-column terms are

$$Y(4,1) = k_2 a \cdot h^3 / 6 \quad , \quad Y(4,2) = k_3 a \cdot h^3 g / 6 \quad ,$$

$$\text{and} \quad Y(4,m) = k_{m+1} \frac{a \cdot h^3 g^{m-1}}{6 (m-1)!} \quad .$$

If the adjacent rows are to be different by a constant, the following must be true;

$$\frac{Y(2,m)}{Y(2,1)} = \frac{k_m g^{m-1}}{(m-1)!} = \frac{Y(4,m)}{Y(4,1)} = \frac{k_{m+1} g^{m-1}}{k_2 (m-1)!} \quad .$$

This implies that $k_m = k_{m+1} / k_2$, or $k_m = k_2^{m-1}$.

This relation can also be derived from the comparison of the terms between any pair of even-order columns.

The series in t given in Eq.(18) can be written as

$$Y(2,m) = (k_2 g)^{m-1} a \cdot h / (m-1)! \quad .$$

This is the series for an exponential function in t , specifically

$$\left. \frac{\partial y}{\partial x} \right|_{x=0} = \exp(k_2 t) \cdot a \quad . \quad (19)$$

The statement equivalent to Eq.(19) for the heat equation is

$$\left. \frac{\partial y}{\partial x} \right|_{x=0} = \exp(-a^2 t) \cdot a \quad ,$$

so we identify $k_2 = -a^2$. This is not a necessary identification since "a"

is a constant whose value is yet to be determined. It is merely convenient to merge the two constants into one; we will be normalizing the solution functions later.

Now, we will consider the boundary condition $u(2,t) = 0$. First, we rewrite Eq.(17) for just the top-row terms;

$$\frac{(n-1)(n-2)}{h^2} U(n,1) = h^2 U(n-4,1) + \frac{1}{g} U(n-2,2) \quad (20)$$

The last term on the right belongs in the second row, which in turns depends on a term in the third row, etc. Here is the reason why it is so important to show that the rows differ by a constant. The multiplier between the top row and the second row is $k_2 g = -a^2 g$; therefore, Eq.(20) becomes

$$\frac{(n-1)(n-2)}{h^2} U(n,1) = h^2 U(n-4,1) - a^2 U(n-2,1) \quad .$$

This equation has only terms in the top row; therefore, the function therein is not a function of t , and we can write

$$\frac{(n-1)(n-2)}{h^2} \phi(n) = h^2 \phi(n-4) - a^2 \phi(n-2) \quad .$$

The solution function of the problem of Eq.(1) is then of the form

$$y_n(x,t) = \exp(-a^2 t) \cdot \phi_n(ax) \quad (21)$$

Since the rows in the 2-D array of the solution function differ only by constants, it is only necessary to match a single row to the boundary condition $u(2,0) = 0$. Just as in the solution of the heat equation, where $\sin(2a) = 0$, we now have that $\phi_n(2a) = 0$. The function $\phi_n(ax)$ with terms up to the 7-th order is as follows

$$\phi_n(ax) = ax - \frac{a^3 x^3}{6} + \frac{a^5 x^5}{20} + \frac{a^5 x^5}{20} - \frac{13a^3 x^7}{2520} - \frac{a^7 x^7}{2520} + \dots$$

For $n = 21$, it is necessary to calculate up to terms of the 209-th order for convergence of the Taylor series. The boundary condition $u(2,0) = 0$ means that $\phi(2a) = 0$. Therefore, the task is to solve for all the values of "a" which satisfy the above condition. The values of "a" found by Newton iteration are listed in Table II.

These values of "a" are accurate, as given in Table II, to 10^{-15} . They are calculated using extended-precision arithmetic with accuracy limited only by the storage capacity of the computer. The necessity of using extended-precision arithmetic comes about from the need for more accuracy than what is available from double-precision on the CDC-6400 computer. The extended-precision arithmetic program is written in FORTRAN using some of the ideas given by Knuth(10). We have found that it is not advisable to perform binary operations when the language is FORTRAN. Therefore, we used integer numbers

with a base as large as possible. For the CDC-6400, this is 10^{+14} . The extended-precision arithmetic operations are performed on numbers stored in arrays with each location interpreted as one base 10^{14} digit. All operations are integer in nature and similar to primary-school arithmetic. For the sake of speed, we use the first location in each array as the sign of the number, and we search each array for the significant digits and only perform operations on non-zero digits. Except for the sign in the first location of the arrays, positive and negative numbers look exactly alike. Extended-precision arithmetic execution time on the CDC-6400 is about 12-times longer than the execution time for double-precision. This comparison is made using numbers that can be handled by double-precision. Once the numbers exceed the limit of double-precision, there can be no meaningful comparison. A listing of the FORTRAN subroutine for extended-precision arithmetic is available from the author.

The next step in the solution of parabolic PDE's is the orthonormalization of the $\phi_n(ax)$ functions. The orthonormalization of the Taylor series is performed using the Gram-Schmidt method, which has been studied in Chang and Colton(11). Using extended-precision arithmetic for ultimate accuracy, we have found that the ϕ_n -functions are orthogonal in the interval $[0,2]$. So, only normalization is required. The normalization process adjusts for the fact that k_2 is not exactly equal to $(-a^2)$.

Let us summarize the steps in the derivation for the solution function, see Eq.(21).

$$y_n(x,t) = \exp(-a^2 t) \cdot \phi_n(ax)$$

The time-dependent portion of the solution function is $\exp(-a^2 t)$ since the rows of the 2-D array differ by constants, see Eq.(19). We have also shown that $\phi_n(ax)$ is a function of x only, see the discussion following Eq.(20). Then, we have constructed the Taylor series terms for $\phi_n(ax)$ and solved for the values of "a" which satisfy $u(0,t) = u(2,t) = 0$, see Table II. The ϕ_n -functions form an orthogonal set on the interval $[0,2]$. And we have normalized them to form an orthonormal set.

The remaining task in the solution of Eq.(1) is to find the Fourier-type approximation for the initial condition using the ϕ_n -functions. Just as in the heat equation solution, we will find the final n solution as the linear combination of $y_n(x,t)$,

$$u(x,t) = \sum_{n=1}^{\infty} c_n \cdot y_n(x,t) \quad , \quad (22)$$

where the coefficients c_n are found by the inner product

$$c_n = \int_0^2 \phi_n(x) \cdot u(x,0) dx \quad .$$

In the next section, we will apply this Taylor series method to three sample problems.

Table II. The Functional Constants for $\phi_n(2a) = 0$.

n	Value of "a"		
1	1.87873	17204	86263
2	3.34204	70011	74951
3	4.85076	94487	09503
4	6.38803	89296	70844
5	7.93823	64977	79745
6	9.49515	38510	55045
7	11.05597	96202	17529
8	12.61927	20295	99243
9	14.18421	87256	09143
10	15.75032	80128	90541
11	17.31728	51600	97127
12	18.88487	94810	13789
13	20.45296	46754	95658
14	22.02143	60050	07426
15	23.59021	65254	73077
16	25.15924	84480	76606
17	26.72848	75315	53938
18	28.29789	93343	11262
19	29.86745	66448	47917
20	31.43713	76800	28297
21	33.00692	47963	29488

3. NUMERICAL EXAMPLES. The three sample problems to be solved differ only in their initial conditions. This permits us to use the same ϕ_n -functions (which are dependent only on the PDE and the boundary conditions) for the solution of increasingly more difficult problems leading up to the problem of Eq.(1).

EXAMPLE 1: $u_{xx} - x^2 \cdot u = u_t$, satisfying $u(0,t) = u(2,t) = 0$, and $u(x,0) = 2x - x^2$, in the interval $[0,2]$. The coefficients c_n calculated for this problem are listed in Table III. Only 13 coefficientsⁿ are listed because sufficient accuracy is obtained at that point. Inclusion of higher-order solution functions would tend to degrade the accuracy, because we did not use extended-precision in this part of the solution. The error given in Table III is the difference between the given initial condition, $u(x,0) = 2x - x^2$, and the solution at the point specified. This is the largest error found for each of the approximate solutions.

Table III. Coefficients c_n for Example 1.

n	c_n	error	at x
1	1.02735	0.33	1.9
2	-.09380	0.19	1.9
3	.04810	0.075	1.9
4	-.00335	0.065	1.9
5	.00900	0.032	1.9
6	-.00042	0.030	1.9
7	.00314	0.015	1.9
8	-.00010	0.015	1.9
9	.00145	0.0072	1.9
10	-.00003	0.0070	1.9
11	.00079	0.0029	1.9
12	-.00001	0.0028	1.9
13	.00048	0.0010	1.8

EXAMPLE 2: $u_{xx} - x^2 \cdot u = u_t$, satisfying $u(0,t) = u(2,t) = 0$, and $u(x,0) = x$ on $[0,1]$, and $u(x,0) = 1 - x$ on $[1,2]$. This is a triangular function whose derivatives at $x=1$ do not exist. The coefficients c_n for this problem are listed in Table IV. All twenty-one solution functionsⁿ are needed in this solution because of the discontinuity at the peak of the triangle. The error given in Table IV is the difference between the initial condition $u(1,0) = 1$ and the approximate solution at $x=1$.

Table IV. Coefficients c_n for Example 2.

n	c_n	error at $x=1$
1	.80734	0.19
2	-.08126	0.18
3	-.08203	0.099
4	.00379	0.099
5	.03237	0.067
6	-.00141	0.067
7	-.01641	0.050
8	.00050	0.050
9	.00999	0.040
10	-.00027	0.040
11	-.00668	0.034
12	.00015	0.034
13	.00479	0.029
14	-.00010	0.029
15	-.00360	0.025
16	.00006	0.025
17	.00280	0.022
18	-.00005	0.022
19	-.00224	0.020
20	.00003	0.020
21	.00209	0.018

EXAMPLE 3: $u_{xx} - x^2 \cdot u = u_t$, satisfying $u(0,t) = u(2,t) = 0$, and $u(x,0) = 1$.

This is the problem stated at the beginning of this discussion; the initial and boundary conditions are inconsistent at $x=0$ and at $x=2$. The coefficients c_n calculated for this problem are listed in Table V. All twenty-one of the solution functions are used in this solution; higher-order functions would have improved the accuracy. However, it should be pointed out that since the ϕ_n -functions are orthogonal and normalized, the solutions as listed in Table V are complete up to the order given. Furthermore, since the higher-order functions have faster exponential decay constants than the low-order functions, the errors given in Table V quickly disappear for small values of time. The error given in Table V is the difference between the initial condition $u(x,0) = 1$ and the approximate solution at $x = 0.05$.

We have chosen to tabulate the error at $x = 0.05$, because this point is only 2.5% of the entire range of the problem. It is true that there is much error at this point with even 21 ϕ_n -functions. However, this error exists only at $t=0$.

Table V. Coefficients c_n for the Problem of Eq.(1).

n	c_n	error at $x=0.05$
1	1.26588	0.88
2	-.09146	0.90
3	.43438	0.80
4	-.01233	0.80
5	.25678	0.70
6	-.00362	0.70
7	.18266	0.61
8	-.00152	0.61
9	.14183	0.52
10	-.00078	0.52
11	.11594	0.43
12	-.00045	0.43
13	.09806	0.34
14	-.00028	0.34
15	.08496	0.27
16	-.00019	0.27
17	.07495	0.19
18	-.00013	0.19
19	.06706	0.13
20	-.00042	0.13
21	.06151	0.066

4. CONCLUSIONS. We have developed a Fourier-type solution of parabolic partial differential equations using two-dimensional Taylor series. The attraction of this method is in its simplicity and the fact that the solution functions can be orthonormalized. Thus, it is possible to ignore the high-order functions if one is not overly concerned with the accuracy at $t=0$.

The solution time on a CDC-6400 computer is divided approximately as follows; 45 seconds for the calculation of the constants "a" given in Table II (up to 15 Newton iterations were required for convergence), 4 seconds to normalize the ϕ_n -functions, and 3 seconds to calculate the c_n coefficients.

5. REFERENCES.

- (1) A. Gibbons, "A program for the automatic integration of differential equations using the method of Taylor series," *Comp. J.*, 3(1960), p108.
- (2) R. D. Richtmeyer, "Detached-shock calculation by power series," AEC R&D Report NYU-7972, New York Univ., New York, (1959).
- (3) E. Fehlberg, "Numerical integration of differential equations by power series expansions," NASA Report TN D-2356, (1964).
- (4) J. G. Hartwell, "Simultaneous integration of n-bodies by analytic continuation with recursively formed derivatives," *J. Astro. Sci.*, 14(1967), p173.
- (5) J. A. Leavitt, "Methods and applications of power series," *Math. Comp.*, 20(1966), p46.
- (6) D. Barton, I. M. Willers, and R. V. M. Zahar, "The automatic solution of systems of ordinary differential equations by the method of Taylor series," *Comp. J.*, 14(1972), p243.
- (7) Y. F. Chang, "Automatic solution of differential equations," *Lecture Notes in Math.* #430, D. L. Colton and R. P. Gilbert, ed, Springer-Verlag, (1974), p61.
- (8) Y. F. Chang, "Solution of parabolic partial differential equations," *Proc. Sixth Manitoba Conf. Num. Math. & Comp.*, (1976).
- (9) Y. F. Chang, "Taylor series solution of linear partial differential equations," presented at SIAM 1976 Fall Meeting, Atlanta, Ga., (1976).
- (10) D. E. Knuth, The Art of Computer Programming, v.2, Seminumerical Algorithms, Addison-Wesley, Reading, Mass. (1969), pp229-279.
- (11) Y. F. Chang and David Colton, "The numerical solution of parabolic partial differential equations by the method of integral operators," to appear *Intl. Jour. Comp. Math.*

A "SINC-GALERKIN" METHOD OF SOLUTION OF BOUNDARY
VALUE PROBLEMS*

Frank Stenger**
University of British Columbia, Vancouver, B. C., Canada

ABSTRACT

This paper illustrates the application of a "Sinc-Galerkin" method to the approximate solution of linear and nonlinear second order ordinary differential equations, and to the approximate solution of some linear elliptic and parabolic partial differential equations in the plane. The method is based on approximating functions and their derivatives by use of the Whittaker cardinal function. The DE is reduced to a system of algebraic equations via new accurate explicit approximations of the inner products, the evaluation of which does not require any numerical integration. Using n function evaluations the error in the final approximation to the solution of the DE is $O(e^{-cn^{1/2d}})$ where c is independent of n , and d denotes the dimension of the region on which the DE is defined.

*Research supported by NRC Grants #67-3973, 67-3990, and 67-9239 at the University of British Columbia, and by U.S. Army Research Contract #DAAG-29-76-G-0210.

**Department of Mathematics, University of Utah, Salt Lake City, Utah 84112, is the permanent address of the author.

1. Introduction and Summary

The function $\text{sinc}(x)$ is defined on the real line by

$$(1.1) \quad \text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

The Whittaker cardinal function of an arbitrary function f is defined for any $h > 0$ by

$$(1.2) \quad C(f, h, x) = \sum_{k=-\infty}^{\infty} f(kh) \text{sinc}\left[\frac{x-kh}{h}\right], \quad h > 0,$$

whenever this series converges.

The approximation of f using a finite number of terms of (1.2) has been extensively studied. The paper [7] contains a review of the properties of $C(f, h, x)$ which were discovered by E.T. Whittaker [14], J.M. Whittaker [15], Hartly [4], Nyquist [8] and Shannon [11]. In [12] new approximations are derived by means of $C(f, h, x)$, for interpolating, integrating and approximating the Fourier (over $(-\infty, \infty)$ only) and Hilbert transforms over $(-\infty, \infty)$, $(0, \infty)$ and $(-1, 1)$. In [6] the function $C(f, h, x)$ is used to obtain formulas for approximating the derivatives of functions over $(-\infty, \infty)$, $(0, \infty)$ and $(-1, 1)$.

In the present paper we use results of [6, 12] to derive basis functions $\{\psi_k\}$ for Galerkin schemes of solving second order problems, and we derive explicit and highly accurate expressions for inner products such as $(f \frac{d^2 u}{dx^2}, \psi_k)$, $(f \frac{du}{dx}, \psi_k)$, (fu, ψ_k) . All of these are expressed in terms of the function values of u , and not the derivatives of u . We then study the application of the derived

approximations on the approximate solution of some ordinary and partial differential equations, via the Galerkin method. The combined method thus yields a system of algebraic equations, without the use of any quadrature.

Let us briefly compare the present method of approximate solution of linear differential equations with currently popular finite difference methods, or with finite element methods that use piecewise linear elements. The finite difference or finite element methods lead to a sparse system of equations. The use of n solution evaluations usually leads to a linear system of n algebraic equations having non-singular coefficient matrix. The error in the resultant approximate solution is $O(n^{-p})$, where p is usually 1 or 2. The "Sinc-Galerkin" method of the present paper also leads to a system of order n on the basis of n solution evaluations. This system has a nonsingular full matrix. The error in the

resulting approximate solution is $O(e^{-cn^{1/(2d)}})$, where d denotes the dimension of the problem. The advantage of the present method is that due to its rapid convergence it does not require the solution of a very large system of equations in order to achieve a desired accuracy, if more than two significant figures of accuracy are required in the approximate solution. In addition, the rate of convergence of the present method is the same, regardless of possible singularities of the solution of an equation on the boundary of the region.

The approximate methods of [6,12] have previously been effectively applied to the approximate solution of integral equations via Galerkin-

type methods in [2,9,10]. In [5] an effective Galerkin-type method is derived which uses approximations derived in [12] to obtain an approximate solution to the problem

$$(1.3) \quad y'' = y - y^3/x^2, \quad y(0) = y(\infty) = 0$$

via the minimization of a certain nonlinear functional. In all of the above cases the error of an n -point approximate solution is $O(e^{-cn^{1/2}})$.

In Sec.2 of the present paper we review the relevant known approximation properties of Whittaker's cardinal function, and we then use these to derive explicit approximate inner products, in general as well as for the important special cases of the intervals $[0,1]$, $[-1,1]$, $[0,\infty]$ and $[-\infty,\infty]$. In Sec.3 we illustrate the application of the previously derived formulas to the approximate solution of some simple "model" problems, such as $u'' = -2$, $u'' = u - u^3/x^2$, $u_t = u_{xx}$ and $u_{xx} + u_{yy} = f$, with appropriate boundary conditions. In Sec.4 we carry out an error analysis, proving the $O(e^{-cn^{1/(2d)}})$ rate of convergence referred to above.

2. Preliminaries and Fundamentals

In this section we shall recall some known properties [18] and derive some new properties of Whittaker's cardinal function, which we shall require in this paper.

Definition 2.1. Let R denote the real line, C the complex plane, and let $B(h)$ denote the family of all functions defined on C that are entire, such that $f \in L^2(R)$ and such that

$$(2.1) \quad |f(z)| \leq C e^{\pi|y|/h}, \quad z = x + iy \in C,$$

for some constant C . Set

$$(2.2) \quad S(j, h)(x) = \text{sinc}\left[\frac{x-jh}{h}\right]$$

and

$$(2.3) \quad \delta_{jk}^{(n)} = S^{(n)}(j, 1)(k) = \left(\frac{d}{dx}\right)^n S(j, 1)(x) \Big|_{x=k}.$$

In particular, we have

$$(2.4) \quad \left\{ \begin{array}{l} \delta_{jk}^{(0)} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases} \\ \delta_{jk}^{(1)} = \begin{cases} 0 & \text{if } j = k \\ \frac{(-1)^{k-j}}{k-j} & \text{if } j \neq k \end{cases} \\ \delta_{jk}^{(2)} = \begin{cases} -\frac{\pi^2}{3} & \text{if } j = k \\ \frac{-2(-1)^{k-j}}{(k-j)^2} & \text{if } j \neq k \end{cases} \end{array} \right.$$

Theorem 2.2 [7]: Let $f \in B(h)$. Then $f(z) = C(f, h, z)$. Moreover

$$(2.5) \quad f(z) = \int_{-\frac{\pi}{h}}^{\frac{\pi}{h}} g(t) e^{izt} dt \quad \text{for some } g \in L^2(-\frac{\pi}{h}, \frac{\pi}{h});$$

$$(2.6) \quad f(z) = \frac{1}{h} \int_{\mathbb{R}} \text{sinc}\left[\frac{z-t}{h}\right] f(t) dt;$$

$$(2.7) \quad \int_{\mathbb{R}} |f(x)|^2 dx = h \sum_{k=-\infty}^{\infty} |f(kh)|^2$$

and the sequence $\{h^{-\frac{1}{2}} S(k, h)\}_{k=-\infty}^{\infty}$ is therefore a complete orthonormal sequence in $B(h)$;

$$(2.8) \quad f \in B(h) \Rightarrow f' \in B(h).$$

Theorem 2.3: Let $\delta_{jk}^{(n)}$ be defined as in (2.3). Then

$$(2.9) \quad \int_{\mathbb{R}} \left\{ \text{sinc}\left[\frac{x-jh}{h}\right] \right\}^{(n)} \text{sinc}\left[\frac{x-kh}{h}\right] dx = h^{1-n} \delta_{jk}^{(n)},$$

$$n = 0, 1, 2, \dots$$

Proof: Let us set

$$(2.10) \quad f(t) = S(j, h)(t)$$

and let us note that $f \in B(h)$. By Eq. (2.8) it thus follows that

$f^{(n)} \in B(h)$, $n = 0, 1, 2, \dots$. Eq. (2.9) thus follows by taking

$f = S(j, h)^{(n)}$ in (2.6), and noting by (2.3) that

$$(2.11) \quad S^{(n)}(j, h)(kh) = h^{-n} \delta_{jk}^{(n)}.$$

Definition 2.4: Let $d > 0$, and let $B(\mathcal{D}'_d)$ denote the family of all functions f that are analytic in

$$(2.12) \quad \mathcal{D}'_d = \{z = x + iy : |y| < d\},$$

such that

$$(2.13) \quad \int_{-d}^d |f(x+iy)| dy \rightarrow 0 \quad \text{as } x \rightarrow \pm\infty$$

and such that $N(f, \mathcal{D}'_d) < \infty$, where

$$(2.14) \quad N(f, \mathcal{D}'_d) = \lim_{y \rightarrow d^-} \left\{ \int_{\mathbb{R}} |f(x+iy)| dx + \int_{\mathbb{R}} |f(x-iy)| dx \right\}$$

Theorem 2.5 [12]: Let h and d be positive, let $f \in B(\mathcal{D}'_d)$, and let $\varepsilon(f)$ be defined by

$$(2.15) \quad \varepsilon(f)(x) = f(x) - C(f, h, x), \quad x \in \mathbb{R}.$$

Then

$$(2.16) \quad \varepsilon(f)(x) = \frac{\sin(\pi x/h)}{2\pi i} \int_{\mathbb{R}} \left[\frac{f(t-id^-)}{(t-x-id)\sin[t-id]\pi/h} - \frac{f(t+id^-)}{(t-x+id)\sin[t+id]\pi/h} \right] dt$$

Moreover

$$(2.17) \quad \|\varepsilon(f)\|_{\infty} = \sup_{x \in \mathbb{R}} |\varepsilon(f)(x)| \leq \frac{N(f, \mathcal{D}'_d)}{2\pi d \sinh(\pi d/h)}.$$

Definition 2.6: Let \mathcal{D} be a simply-connected domain in the complex plane \mathbb{C} , and let \mathcal{D}'_d be defined as in (2.12). Let ϕ be a conformal map of \mathcal{D} onto \mathcal{D}'_d , and let $\psi = \phi^{-1}$ denote the inverse map. Let $a = \psi(-\infty)$ and $b = \psi(\infty) \neq a$ be boundary points of \mathcal{D} , and let us take

$$(2.18) \quad \Gamma = \{w \in \mathcal{D} : w = \psi(x), \quad -\infty \leq x \leq \infty\}.$$

Let $B(\mathcal{D})$ denote the family of all functions that are analytic in \mathcal{D} , such that for u real

$$(2.19) \quad \int_{\psi(L+u)} |f(z)dz| \rightarrow 0 \quad \text{as } u \rightarrow \pm\infty$$

where

$$(2.20) \quad L = \{iy : -d \leq y \leq d\},$$

and such that

$$(2.21) \quad N(f, \mathcal{D}) = \liminf_{C \rightarrow \partial \mathcal{D}, C \subset \mathcal{D}} \int_C |f(z)dz| < \infty$$

(Note that if $f \in B(\mathcal{D})$, then $f \circ \psi \in B(\mathcal{D}'_d)$.) Set

$$(2.22) \quad x_k = \psi(kh), \quad k = 0, \pm 1, \pm 2, \dots,$$

and let g be a function which is analytic in \mathcal{D} , whose properties we shall determine in the sequel. Finally, we set

$$(2.23) \quad S_j(z) = g(z) \operatorname{sinc} \left[\frac{\phi(z) - kh}{h} \right] = g(z) S(j, h) \circ \phi(z).$$

The following result was established in [6].

Theorem 2.7: Let m be a nonnegative integer, and let $f\phi'/g \in B(\mathcal{D})$. Let there exist positive constants $\alpha > 0$, C_0 depending only on m, d and g , C_1 depending only on m and g , and C_2 depending only on m, g and f , such that

$$(2.24) \quad \left| \frac{f(x)}{g(x)} \right| \leq C_2 e^{-\alpha |\phi(x)|} \quad \text{for all } x \in \Gamma$$

$$\begin{aligned}
 (2.25) \quad & \left| \left(\frac{d}{dx} \right)^n S_k(x) \right| \leq C_1 h^{-n} \text{ for all } x \in \Gamma \\
 (2.26) \quad & \left| \left(\frac{d}{dx} \right)^n \left\{ \frac{g(x) \sin[\pi \phi(x)/h]}{\phi(z) - \phi(x)} \right\} \right| \leq C_0 h^{-n} \text{ for all } x \in \Gamma, z \in \partial D
 \end{aligned}
 \left. \vphantom{\begin{aligned} (2.25) \\ (2.26) \end{aligned}} \right\} n=0,1,\dots,m.$$

Then there exists a constant K depending only on m, d, α, g and f such that if $h = [\pi d / (\alpha N)]^{1/2}$ then

$$(2.27) \quad \left| f^{(n)}(x) - \sum_{j=-N}^N \frac{f(x_j)}{g(x_j)} S_j^{(n)}(x) \right| \leq K N^{\frac{n+1}{2}} \exp[-(\pi d \alpha N)^{1/2}]$$

for all $x \in \Gamma$, and for $n = 0, 1, \dots, m$.

Theorem 2.8 [12]: If $f \in B(D)$, then the identity

$$\begin{aligned}
 (2.28) \quad & \frac{f(x)}{\phi'(x)} - \sum_{j=-\infty}^{\infty} \frac{f(x_j)}{\phi'(x_j)} S(j, h) \phi(x) \\
 &= \frac{\sin[\pi \phi(x)/h]}{2\pi i} \int_{\partial D} \frac{f(z) dz}{[\phi(z) - \phi(x)] \sin[\pi \phi(z)/h]}
 \end{aligned}$$

is valid for all $x \in \Gamma$. Moreover

$$\begin{aligned}
 (2.29) \quad & \int_{\Gamma} f(x) dx - h \sum_{j=-\infty}^{\infty} \frac{f(x_j)}{\phi'(x_j)} \\
 &= \frac{1}{2} \int_{\partial D} \frac{\exp\left[\frac{i\pi \phi(z)}{h} \operatorname{sgn} \operatorname{Im} \phi(z)\right]}{\sin\left[\frac{\pi}{h} \phi(z)\right]} f(z) dz
 \end{aligned}$$

The results of this theorem may be conveniently combined with those of the formulas obtained above, to yield explicit approximate expressions for inner products. The results of the following lemma are useful for

bounding the error of these approximate expressions.

Lemma 2.9: If $|\operatorname{Im} z| = d > 0$ and if k is an integer, then

$$(2.30) \quad \frac{1}{2} \left| \frac{\operatorname{sinc}[(z-kh)/h]}{\sin(\pi z/h)} \right| \leq C_1(h,d) \equiv \frac{h}{2\pi d}$$

$$(2.31) \quad \frac{1}{2} \left| \frac{(d/dz)\{\operatorname{sinc}[(z-kh)/h]\}}{\sin(\pi z/h)} \right| \leq C_2(h,d) \equiv \frac{d+(h/\pi)\tanh(\pi d/h)}{2d^2 \tanh(\pi d/h)}$$

$$(2.32) \quad \frac{1}{2} \left| \frac{\frac{d^2}{dz^2}\{\operatorname{sinc}[(z-kh)/h]\}}{\sin(\pi z/h)} \right| \leq C_3(h,d) \equiv \frac{[(2h/\pi)+\pi^2 d/h]d \tanh(\pi d/h)+2d}{2d^3 \tanh(\pi d/h)}$$

$$(2.33) \quad |\sin(\pi z/h)| \geq \sinh(\pi d/h), \quad |\cos(\pi z/h)| \leq \cosh(\pi d/h)$$

Proof: We shall only prove (2.31), since the proofs of the remaining cases are similar, and we omit them. We have

$$w = \frac{d}{dz} \{S(k,h)(z)\} = \frac{\cos[\pi(z-kh)/h]}{z-kh} - \frac{h}{\pi} \frac{\sin[\pi(z-kh)/h]}{(z-kh)^2}$$

Now if $|\operatorname{Im} z| = d$, then $|z-kh| \geq d$, $|\cos[\pi(z-kh)]| \leq \cosh(\pi d/h)$ and $|\sin(\pi z/h)| \geq \sinh(\pi d/h)$; hence

$$\left| \frac{w}{\sin(\pi z/h)} \right| \leq \frac{1}{d \tanh(\pi d/h)} + \frac{h}{\pi d^2} = C_2(h,d).$$

Theorem 2.10: Let $\delta_{jk}^{(n)}$ be defined as in (2.3) and (2.4), let $C_j(h,d)$ be defined as in Lemma 2.9, let x_k be defined as in (2.22), S_k as in (2.23), set $F_k = F(x_k)$ for an arbitrary function F , and let r and f be functions which are analytic in \mathcal{D} .

(a) Let $rfg \in B(\mathcal{D})$. Then

$$(2.34) \quad \left| \int_{\Gamma} r(x) f(x) S_k(x) dx - h \frac{f_k r_k g_k}{\phi_k'} \right| \leq C_1(h, d) N(rfg, \mathcal{D}) e^{-\pi d/h}.$$

(b) Let $[rfg/\phi](x) \rightarrow 0$ as $x \rightarrow a$ and as $x \rightarrow b$ along Γ , and let $(rg)'f$ and $rg\phi'f \in B(\mathcal{D})$. Then

$$(2.35) \quad \left| \int_{\Gamma} r(x) f'(x) S_k(x) dx + h \sum_{j=-\infty}^{\infty} f_j \left\{ \frac{(rg)_j'}{\phi_j'} \delta_{kj}^{(0)} + (rg)_j \frac{\delta_{kj}^{(1)}}{h} \right\} \right| \\ \leq [C_1(h, d) N(f(rg)', \mathcal{D}) + C_2(h, d) N(frg\phi', \mathcal{D})] e^{-\pi d/h}.$$

(c) Let $[frg'/\phi](x)$, $[frg\phi'/\phi](x)$ and $[f'rg/\phi](x) \rightarrow 0$ as $x \rightarrow a$ and as $x \rightarrow b$ along Γ , and let $f(rg)''$, $f[2(rg)'\phi' + rg\phi'']$ and $frg(\phi')^2 \in B(\mathcal{D})$. Then

$$(2.36) \quad \left| \int_{\Gamma} r(x) f''(x) S_k(x) dx - h \sum_{j=-\infty}^{\infty} f_j \left\{ \frac{(rg)_j''}{\phi_j'} \delta_{kj}^{(0)} + \frac{[2(rg)_j'\phi_j' + (rg)_j\phi_j'']}{\phi_j'} \frac{\delta_{kj}^{(1)}}{h} + (rg)_j\phi_j' \frac{\delta_{kj}^{(2)}}{h^2} \right\} \right| \\ \leq [C_1(h, d) N(f(rg)'', \mathcal{D}) + C_2(h, d) N(f[2(rg)'\phi' + rg\phi''], \mathcal{D}) + C_3(h, d) N(frg(\phi')^2, \mathcal{D})] \\ \cdot e^{-\pi d/h}.$$

Proof: We shall only prove the (b)-part of Theorem 2.9, since the proofs of the (a) and (c)-parts are similar.

We find, upon integration by parts, that

$$(2.37) \quad \int_{\Gamma} r(x) f'(x) S_k(x) dx = r(x) f(x) S_k(x) \Big|_a^b \\ - \int_{\Gamma} f(x) [r(x) S_k'(x) + r'(x) S_k(x)] dx$$

The first term on the right-hand side vanishes, by assumption of the (b)-part of the theorem, while by expansion of the second part of (2.37), we have

$$(2.38) \quad \int_{\Gamma} r(x) f'(x) S_k(x) dx \\ = - \int_{\Gamma} f(x) [(rg)'(x) S(k, h) \epsilon \phi(x) + (rg\phi')(x) S'(k, h) \epsilon \phi(x)] dx$$

Hence by replacing f in (2.29) by the integrand on the right-hand side of (2.38), and noting that if $z \in \partial D$, then $|\operatorname{Im} \phi(z)| = d$ and $|\exp[\frac{i\pi}{h} \phi(z) \operatorname{sgn} \operatorname{Im} \phi(z)]| = e^{-\pi d/h}$ we find by (2.29), Lemma 2.9 and Theorem 2.3, that

$$\int_{\Gamma} r(x) f'(x) S_k(x) dx \\ + h \sum_{j=-\infty}^{\infty} f_j \left\{ \frac{(rg)'_j}{\phi'_j} \delta_{kj}^{(0)} + (rg)_j \frac{\delta_{kj}^{(1)}}{h} \right\} \\ \leq e^{-\pi d/h} \int_{\partial D} [C_1(h, d) |[f(rg)'](z)| + C_2(h, d) |(frg\phi')(z)|] |dz|,$$

which is just (2.35).

Theorem 2.11: Let N be a positive integer, α a positive constant, and take $h = [\pi d / (\alpha N)]^{\frac{1}{2}}$.

(a) Under the assumptions of Theorem 2.9 (a),

$$(2.39) \quad \left| \int_{\Gamma} r(x) f(x) S_k(x) dx - h \frac{f_k r_k g_k}{\phi_k} \right| \leq \frac{K_1}{N^{\frac{1}{2}}} e^{-(\pi \alpha N)^{\frac{1}{2}}},$$

where K_1 depends only on f, r, g, d and α ;

(b) If $|[rgf](x)| \leq K'_2 \exp[-\alpha|\phi(x)|]$ on Γ , then under the assumptions of Theorem 2.9 (b),

$$(2.40) \quad \left| \int_{\Gamma} r(x) f'(x) S_k(x) dx + h \sum_{j=-N}^N f_j \left\{ \frac{(rg)_j'}{\phi_j'} \delta_{kj}^{(0)} + (rg)_j \frac{\delta_{kj}^{(1)}}{h} \right\} \right| \\ \leq K_2 e^{-(\pi d \alpha N)^{\frac{1}{2}}}, \quad k = -N, -N+1, \dots, N,$$

where K_2 depends only on f, r, g, d and α ;

(3) If $[f\{2(rg)' + rg\phi''/\phi'\}](x)$ and $[rgf\phi'](x)$ are bounded by $K'_3 \exp[-\alpha|\phi(x)|]$ on Γ , then under the assumptions of Theorem 2.9 (c),

$$(2.41) \quad \left| \int_{\Gamma} r(x) f''(x) S_k(x) dx - h \sum_{j=-N}^N f_j \left\{ \frac{(rg)_j''}{\phi_j'} \delta_{kj}^{(0)} + \frac{[2(rg)_j' \phi + (rg)_j \phi'']}{\phi_j'} \frac{\delta_{kj}^{(1)}}{h} + (rg)_j \phi_j' \frac{\delta_{kj}^{(2)}}{h^2} \right\} \right| \\ \leq K_3 N^{\frac{1}{2}} e^{-(\pi d \alpha N)^{\frac{1}{2}}}, \quad k = -N, -N+1, \dots, N,$$

where K_3 depends only on f, r, g, d and α .

Proof: The proof is similar to that of Theorem 8.1 of [12], and we omit it.

The results of Theorem 2.10 are especially suited to the solution of linear differential equations via a Galerkin method, for which the functions $\{S_k\}$ are the approximating basis functions. We remark that we could have obtained alternate expressions of $\int_{\Gamma} r(x) f^{(n)}(x) S_k(x) dx$, by combining equations (2.29) and (2.27), i.e., if $rgf^{(n)} \in B(\mathcal{D})$, then by Eq. (2.29)

$$(2.42) \quad \left| \int_{\Gamma} r(x) f^{(n)}(x) S_k(x) dx - \frac{h r_k g_k f^{(n)}(x_k)}{\phi_k'} \right| \\ \leq C_1(h, d) N(rgf^{(n)}, \mathcal{D}) e^{-\pi d/h}$$

and we could now use (2.29) to approximate $f^{(n)}(x_k)$ on Γ . However, the resulting expressions are not as accurate as those of Theorem 2.10. Nevertheless the pair of equations (2.27) and (2.29) do form a powerful combination for purposes of solving nonlinear equations. For example, if $G \in B(\mathcal{D})$, where $G = G(x, f(x), f'(x))$, then

$$(2.43) \quad \left| \int_{\Gamma} G(x, f(x), f'(x)) S_k(x) dx - h \frac{G(x_k, f(x_k), f'(x_k))}{\phi'(x_k)} g(x_k) \right|$$

$$\leq C_1(h, d) N(G, \mathcal{D}) e^{-\pi d/h};$$

if the conditions of Theorem 2.7 are satisfied for $m = 1$, we may now replace $f'(x_k)$ in (2.43) by the approximation

$$(2.44) \quad f'(x_k) \cong \sum_{j=-N}^N \frac{f_j}{g_j} [g_j' \delta_{jk}^{(0)} + g_j \phi_j' \frac{\delta_{jk}^{(1)}}{h}]$$

given by Eq. (2.27).

The approximating expressions of Theorem 2.10 may be more compactly expressed by means of matrices. To this end, let $m = 2N + 1$, and let

\underline{S}_m and \underline{f}_m be column vectors defined by

$$(2.45) \quad \underline{S}_m(x) = \begin{pmatrix} S_{-N}(x) \\ S_{-N+1}(x) \\ \vdots \\ S_N(x) \end{pmatrix}, \quad \underline{f}_m = \begin{pmatrix} f_{-N} \\ f_{-N+1} \\ \vdots \\ f_N \end{pmatrix}$$

Corresponding to a function $u = u(x)$, let $\underline{A}_m(u)$ denote a diagonal

matrix, whose diagonal elements are $u(x_{-N}), u(x_{-N+1}), \dots, u(x_N)$ and whose off-diagonal elements are zero. Let $\tilde{I}_m^{(1)}$ and $\tilde{I}_m^{(2)}$ denote the matrices

$$(2.46) \quad \tilde{I}_m^{(1)} = \begin{bmatrix} 0 & 1 & -\frac{1}{2} & +\frac{1}{3} & \dots & -\frac{1}{2N} \\ -1 & 0 & 1 & -\frac{1}{2} & \dots & \frac{1}{2N-1} \\ & & \dots & & & \\ \frac{1}{2N} & -\frac{1}{2N-1} & \frac{1}{2N-2} & -\frac{1}{2N-3} & \dots & 0 \end{bmatrix}$$

$$(2.47) \quad \tilde{I}_m^{(2)} = \begin{bmatrix} -\frac{\pi^2}{3} & \frac{2}{1^2} & -\frac{2}{2^2} & \dots & \frac{-2}{(2N)^2} \\ \frac{2}{1^2} & -\frac{\pi^2}{3} & \frac{2}{1^2} & \dots & \frac{2}{(2N-1)^2} \\ & & \dots & & \\ -\frac{2}{(2N)^2} & \frac{2}{(2N-1)^2} & -\frac{2}{(2N-2)^2} & \dots & -\frac{\pi^2}{3} \end{bmatrix}$$

With this notation, Eqns. (2.39), (2.40) and (2.41) take the approximating form

$$(2.48) \quad \begin{aligned} \int_{\Gamma} r(x) f(x) S_m(x) dx &= h \tilde{A}_m \left(\frac{rg}{\phi'} \right) f_m \\ \int_{\Gamma} r(x) f'(x) S_m(x) dx &= -h \left[\tilde{A}_m \left(\frac{(rg)'}{\phi'} \right) + \frac{1}{h} \tilde{I}_m^{(1)} \tilde{A}_m(rg) \right] f_m \\ \int_{\Gamma} r(x) f''(x) S_m(x) dx &= h \left[\tilde{A}_m \left(\frac{(rg)''}{\phi'} \right) + \frac{1}{h} \tilde{I}_m^{(1)} \tilde{A}_m(2(rg) + rg\phi''/\phi') \right. \\ &\quad \left. + \frac{1}{h^2} \tilde{I}_m^{(2)} \tilde{A}_m(rg\phi') \right] f_m \end{aligned}$$

Contrary to the case of matrices that arise in the solution of differential equations by finite difference or finite element methods, relatively little is known about the matrices $\tilde{I}_m^{(1)}$ and $\tilde{I}_m^{(2)}$ in (2.46) and (2.47). To this end, we record two simple observations.

Theorem 2.12. The matrix $\tilde{I}_m^{(1)}$ in (2.46) is a skew symmetric matrix of odd order, $m = 2N + 1$ and it is therefore singular. The matrix $\tilde{I}_m^{(2)}$ is a non-singular symmetric matrix of order $m = 2N + 1$.

Proof: (due to A. Adler at U.B.C.) Expanding $\tilde{I}_m^{(2)}$, we have $-3^m \det \tilde{I}_m^{(2)} = \pi^{4N+2} + c_1 \pi^{4N} + c_2 \pi^{4N-2} + \dots + c_{2N}$ where the c_j are rational. The number π^2 is a transcendental number, whereas the vanishing of $\det \tilde{I}_m^{(2)}$ would imply that π^2 is an algebraic number.

We close this section with a derivation of the formulas of Theorem 2.10 for the case of the important intervals $[0,1]$, $[-1,1]$, $[0,\infty]$, and $[-\infty,\infty]$.

Ex.1: $\Gamma = [0,1]$. In this case

$$(2.49) \quad \begin{cases} \phi(z) = \log\left(\frac{z}{1-z}\right) , & \phi'(z) = \frac{1}{z(1-z)} , \\ \mathcal{D} = \{z: |\arg\left(\frac{z}{1-z}\right)| < d\} . \end{cases}$$

Let us assume that the coefficients r of a second order equation are analytic and bounded in \mathcal{D} , and that the same is true of r' and r'' . It is then convenient to take

$$(2.50) \quad g(x) = \frac{1}{\phi'(x)} = x(1-x) .$$

The conditions of Theorem 2.10 are satisfied if f is analytic and bounded

on \mathcal{D} , and if on $[0,1]$, $|f(x)| \leq C[x(1-x)]^\alpha$, where C and α are positive constants. If f does not vanish at 0 and 1, we replace f by F in the differential equation, where

$$(2.51) \quad F(x) = f(x) - a(1-x) - bx$$

and where $a = f(0)$, $b = f(1)$. The basis functions are

$$(2.52) \quad \{S_k(x)\}_{k=-N}^N = \{x(1-x)S(k,h) \circ \phi(x)\}_{k=-N}^N.$$

To these, it may be necessary to adjoin $1-x$ if a is unknown, and x if b is unknown. Differentiating g and ϕ' , we get

$$(2.53) \quad g'(x) = 1 - 2x, \quad g''(x) = -2, \quad \phi''(x) = -\frac{1-2x}{x^2(1-x)^2}.$$

Hence

$$(2.54) \quad \begin{cases} (rg)(x) = x(1-x)r(x) & \left(\frac{rg}{\phi'}\right)(x) = x^2(1-x)^2r(x) \\ \left(\frac{(rg)'}{\phi'}\right)(x) = x(1-x)[x(1-x)r'(x) + (1-2x)r(x)] \\ \left(\frac{(rg)''}{\phi'}\right)(x) = x(1-x)[x(1-x)r''(x) + 2(1-x)r'(x) - 2r(x)] \\ \left(\frac{2(rg)'\phi' + rg\phi''}{\phi'}\right)(x) = 2x(1-x)r'(x) + (1-2x)r(x) \\ (rg\phi')(x) = r(x) \end{cases}$$

Hence we get the approximations (2.48), in which $x_k = \frac{1}{2} + \frac{1}{2} \tanh(kh/2)$.

Ex.2. $\Gamma = [-1,1]$. In this case

$$\phi(z) = \log\left(\frac{1+z}{1-z}\right), \quad \phi'(z) = \frac{2}{1-z^2}$$

(2.55)

$$\mathcal{D} = \{z: |\arg(\frac{1+z}{1-z})| < d\}.$$

Under assumptions on r similar to those of Ex. 1, we take

$$g(x) = \frac{1}{\phi'(x)} = \frac{1}{2}(1-x^2)$$

(2.56)

The conditions of Theorems 2.9 and 2.10 are satisfied if f is analytic and bounded on \mathcal{D} , and if on $(-1,1)$, $|f(x)| \leq C(1-x^2)^\alpha$, where C and $\alpha > 0$. If f does not vanish on -1 and 1 , we set $f = F + p$ in the differential equation, where

$$p(x) = a \frac{1-x}{2} + b \frac{1+x}{2}$$

(2.57)

and where $a = f(-1)$, $b = f(1)$. The basis functions are

$$\{S_k(x)\}_{k=-N}^N = \left\{\frac{1}{2}(1-x^2)S(k,h) \circ \phi(x)\right\}_{k=-N}^N;$$

(2.58)

to these it may be necessary to adjoin $(1-x)/2$ and/or $(1+x)/2$ if a and/or b are unknown. Differentiating g and ϕ' , we get

$$g'(x) = -x, \quad g''(x) = -1, \quad \phi''(x) = x\left(\frac{2}{1-x^2}\right)^2,$$

(2.59)

so that

$$(2.60) \quad \begin{cases} (rg)(x) = \frac{1}{2}(1-x^2)r(x) ; & (\frac{rg}{\phi})(x) = (\frac{1-x^2}{2})^2 r(x) \\ (\frac{(rg)'}{\phi})(x) = (\frac{1-x^2}{2})^2 r'(x) - x(\frac{1-x^2}{2})r(x) \\ (\frac{(rg)''}{\phi})(x) = (\frac{1-x^2}{2})^2 r''(x) - x(1-x^2)r'(x) - (\frac{1-x^2}{2})r(x) \\ (2(rg)' + rg\phi''/\phi')(x) = (1-x^2)r'(x) - xr(x) \\ (rg\phi')(x) = r(x) \end{cases}$$

Hence, with $x_k = \tanh(kh/2)$, the approximations of Theorem 2.10 take the form (2.48).

Ex.3: The case $\Gamma = [0, \infty]$. In this case

$$(2.61) \quad \phi(z) = \log z, \quad \phi'(z) = \frac{1}{z}, \quad \mathcal{D} = \{z: |\arg z| < d\}.$$

Suppose that the coefficients r and the derivatives of r are analytic and bounded in \mathcal{D} . If on \mathcal{D} , $|f(z)| \leq C|z|^\alpha/(1+|z|)^{2\alpha}$ where C, α are positive then it is convenient to take $g(z) = z/(1+z)^2$, in order that the conditions of Theorems 2.9 and 2.10 are satisfied. However, if $|f(z)| \leq C|z|^\alpha/(1+|z|)^{2+\alpha}$ in \mathcal{D} , where C and α are positive, then it is possible to choose a simpler form for g , and $S_k(x)$, namely

$$(2.62) \quad g(x) = \frac{1}{\phi'(x)} = x ; \quad S_k(x) = g(x)S(k,h) \circ \phi(x).$$

In this latter case

$$(2.63) \quad \begin{cases} (rg)(x) = xr(x) & \left(\frac{rg}{\phi'}\right)(x) = x^2 r(x) \\ \left(\frac{(rg)'}{\phi'}\right)(x) = x^2 r'(x) + xr(x) & ; \quad \left(\frac{(rg)''}{\phi'}\right)(x) = x^2 r''(x) + 2xr'(x) \\ (2(rg)' + rg\phi''/\phi')(x) = 2xr'(x) + r(x) \\ (rg\phi')(x) = r(x) \end{cases}$$

The approximations now take the form (2.48), in which $x_k = e^{kh}$.

If f is merely bounded on \mathcal{D} , and if $d = \lim_{(x \rightarrow \infty)} x^2 f'(x)$, then we replace f by F in the differential equation, where

$$(2.64) \quad f(x) = F(x) + \frac{a}{1+x} + \frac{xb}{1+x} + \frac{xc}{(1+x)^2}$$

where

$$(2.65) \quad a = f(0), \quad b = f(\infty), \quad c = b - a - d.$$

If the limit $\lim_{(x \rightarrow \infty)} x^2 f'(x)$ does not exist, it may be better to take $g(x) = x/(1+x)$ or $g(x) = x/(1+x)^2$, depending upon the problem.

Ex.4: The case $\Gamma = [-\infty, \infty]$. In this case

$$(2.66) \quad \phi(z) = z, \quad \phi'(z) = 1,$$

and $\mathcal{D} = \mathcal{D}'_d$ (see Eq. (2.12)). If the coefficients r of the differential equation are analytic and bounded in \mathcal{D}'_d , and if $f \in B(\mathcal{D}'_d)$, we simply take

$$(2.67) \quad g'(x) = 1, \quad \{S_k(x)\}_{k=-N}^N = \{S(k,h)(x)\}_{k=-N}^N$$

in order that the conditions of Theorem 2.9 become satisfied, and provided

that f vanishes at $\pm\infty$. The conditions of Theorem 2.10 also become satisfied if $|f(x)| \leq Ce^{-\alpha|x|}$ on Γ . Then $x_k = kh$, and

$$(2.68) \quad \begin{cases} (rg)(x) = r(x), & (\frac{rg}{\phi'}) (x) = r(x), & (\frac{(rg)'}{\phi'}) (x) = r'(x) ; \\ (\frac{(rg)''}{\phi'}) (x) = r''(x), & (2(rg)' + rg\phi''/\phi') (x) = 2r'(x) \\ (rg\phi') (x) = r(x) . \end{cases}$$

The approximating equations again take the form (2.48).

If f does not vanish at $\pm\infty$, we replace f by F , where

$$(2.69) \quad F(x) = f(x) - \frac{1}{e^{cx} + e^{-cx}} [e^{-cx}f(-\infty) + e^{cx}f(\infty)]$$

and where $0 < c < \pi/(2d)$.

3. Examples of Applications

In this section we shall illustrate the application of the formulas developed in Sec.2, on the solution of some simple ordinary and partial differential equations.

Ex.1: Consider the simple problem

$$(3.1) \quad f_{xx}(x) = -2, \quad 0 < x < 1; \quad f(0) = f(1) = 0$$

This has the solution $f(x) = x(1-x)$. By taking $r(x) = 1$ in (2.54) and combining with (2.48), we arrive at the system of equations

$$(3.2) \quad h[-2\tilde{A}_m(x(1-x)) + \frac{1}{h}\tilde{I}_m^{(1)}\tilde{A}_m(x(1-x)) + \frac{1}{h^2}\tilde{I}_m^{(2)}]f_m = -2h\tilde{A}_m(x^2(1-x)^2)e$$

where $e = (1, 1, \dots, 1)^T$, T denoting the transpose. Solving this system for the case $N = 2$, $h = x_k = \frac{1}{2} + \frac{1}{2} \tanh(kh/2)$, we get the approximations

$$f_{-2} = .00575, \quad f_{-1} = .0856, \quad f_0 = .2495, \quad f_1 = .0856, \quad f_2 = .00575.$$

These are accurate to 3 significant figures.

Ex.2. $f'' = f - f^3/x^3$, $f(0) = f(\infty) = 0$. This problem was solved by different procedures in [1] and [5]. By taking $x_k = e^{kh}$ and combining (2.63) and (2.48), we get the approximating system

$$(3.3) \quad [\tilde{I}_m^{(1)} + \frac{1}{h}\tilde{I}_m^{(2)}]f_m = h\tilde{A}_m(x^2)[f_m - e_m]$$

where \underline{e}_m denotes the vector $[x_{-N}^{-2} f_{-N}^3, x_{-N+1}^{-2} f_{-N+1}^3, \dots, x_N^{-2} f_N^3]^T$.

The solution of (3.3) involves the solution of a system of nonlinear equations. By taking $h = 1$, $N = 2$ ($m = 5$) we get the approximations

$$f_{-2} = .0855 \quad f_{-1} = 1.325 \quad f_0 = 5.834 \quad f_1 = 1.114 \quad f_2 = .1132$$

which are accurate to 3 significant figures.

Ex.3.
$$\begin{cases} u_{xx} = u_t, & 0 < x < 1, \quad t > 0 \\ u(x, 0) = \sin \pi x, \quad u(0, t) = u(1, t) = 0. \end{cases}$$

In order to get zero boundary conditions, we set

$$(3.5) \quad u = v + \sin(\pi x) e^{-4t}.$$

This yields the problem

$$(3.6) \quad \begin{cases} v_{xx} - v_t = (\pi^2 - 4) \sin(\pi x) e^{-4t}, & 0 < x < 1, \quad t > 0 \\ v(x, 0) = 0, \quad v(0, t) = v(1, t) = 0. \end{cases}$$

We solve this by taking our approximating basis functions to be

$$(3.7) \quad \begin{cases} S_k(x) = x(1-x) S(k, h) \circ \phi(x), & \phi(x) = \log[x/(1-x)] \\ S_\ell^*(t) = t S(\ell, h) \circ \phi^*(t), & \phi^*(t) = \log t. \end{cases}$$

The problem (3.6) may now readily be reduced to a matrix problem, by proceeding as for (3.2) above. Setting

$$(3.8) \quad \underline{V} = \begin{bmatrix} v_{-N,N} & v_{-N,-N+1} & \dots & v_{-N,N} \\ v_{-N+1,-N} & v_{-N+1,-N+1} & \dots & v_{-N+1,N} \\ & & \dots & \\ v_{N,-N} & v_{N,-N+1} & \dots & v_{N,N} \end{bmatrix}$$

$$(3.9) \quad \underline{B} = -2hA_m(x(1-x)) + I_m^{(1)} A_m(x(1-x)) + \frac{1}{h} I_m^{(2)} \quad (x_k = \frac{1}{2} + \frac{1}{2} \tanh \frac{kh}{2})$$

$$(3.10) \quad \underline{C} = -(hI_m^0 + I_m^{(1)}) A_m(t) \quad (I_m^{(0)} = \text{unit matrix}, \quad t_k = e^{kh})$$

$$(3.11) \quad \underline{D} = hA_m(x^2(1-x)^2), \quad x_k = \frac{1}{2} + \frac{1}{2} \tanh(kh/2)$$

$$(3.12) \quad \underline{E} = hA_m(t^2), \quad t_k = e^{kh}$$

$$(3.13) \quad \underline{F} = \underline{D} \begin{bmatrix} F_{-N,-N} & F_{-N,-N+1} & \dots & F_{-N,N} \\ F_{-N+1,-N} & F_{-N+1,-N+1} & \dots & F_{-N+1,N} \\ & & \dots & \\ F_{N,-N} & F_{N,-N+1} & \dots & F_{N,N} \end{bmatrix}$$

where

$$F_{k\ell} = (\pi^2 - 4) \sin(\pi x_k) e^{-4t_\ell},$$

we arrive at the matrix system

$$(3.14) \quad \underline{BDV} + \underline{VEC} = \underline{F}$$

Eq. (3.14) may be solved by diagonalizing $\underline{\underline{BD}}$ and $\underline{\underline{EC}}$. If

$\lambda_{-N}, \lambda_{-N+1}, \dots, \lambda_N$ and $\mu_{-N}, \mu_{-N+1}, \dots, \mu_N$ denote the eigenvalues of $\underline{\underline{BD}}$ and $\underline{\underline{EC}}$ respectively, obtained by taking $\underline{\underline{X}}^{-1} \underline{\underline{BD}} \underline{\underline{X}}$ and $\underline{\underline{Z}} \underline{\underline{EC}} \underline{\underline{Z}}^{-1}$ via e.g. the method of Galub and Kahan [3], and if $\underline{\underline{G}} = [g_{k\ell}] = \underline{\underline{X}}^{-1} \underline{\underline{F}} \underline{\underline{Z}}^{-1}$, $\underline{\underline{Y}} = [y_{k\ell}] = \underline{\underline{X}}^{-1} \underline{\underline{V}} \underline{\underline{Z}}^{-1}$ then $y_{k\ell} = g_{k\ell} / (\lambda_k + \mu_\ell)$, and $\underline{\underline{V}} = \underline{\underline{X}} \underline{\underline{Y}} \underline{\underline{Z}}$.

By taking $h = 1$, $N = 2$, we get a solution which is accurate to 3 dec. on $[0,1] \times [0,\infty]$.

Ex.4

$$\begin{aligned} u_{xx} + u_{yy} &= -1, \quad (x,y) \in S \equiv [0,1] \times [0,1] \\ (3.15) \quad u &= 0 \quad \text{on } \partial S \end{aligned}$$

Letting B and D be defined as in (3.9) and (3.11) we now get the approximating matrix system

$$(3.16) \quad \underline{\underline{BDU}} + \underline{\underline{UDB}} = -\underline{\underline{DHD}}$$

where $\underline{\underline{U}} = [u_{k\ell}]$, $\underline{\underline{H}} = [h_{k\ell}]$, $h_{k\ell} = 1$. This may now be readily solved via the diagonalization of $\underline{\underline{BD}}$. By taking $N = 2$, $h = 1$, we get a solution which is accurate to 3 dec.

4. Error Analysis

For sake of simplicity, we shall restrict ourselves to the simpler case of the second order problem

$$(4.1) \quad u'' + f(x, u) = 0, \quad u(0) = u(1) = 0$$

The analysis for the case of other ordinary or partial differential equations is somewhat more complicated, but may be carried out similarly.

Throughout this section $\alpha, C_1, C_2, \dots, C_{16}$ denote positive constants, and $h = [\pi d / (\alpha N)^{\frac{1}{2}}]$.

In the notation of the previous sections, we take $\phi(z) = \log[z/(1-z)]$, and we take the domain of analyticity to be $\mathcal{D} = \{z: |\arg[z/(1-z)]| < d\}$. We shall assume that (4.1) has a (locally) unique solution u_0 which is analytic and bounded in \mathcal{D} and which satisfies the inequality

$$(4.2) \quad |u_0(x)| \leq C_1 x^\alpha (1-x)^\alpha, \quad 0 \leq x \leq 1$$

Definition 4.1. Let $M(d, \alpha)$ denote the family of all functions v that are analytic in \mathcal{D} , such that

$$(4.3) \quad \begin{cases} v(0) = v(1) = 0 \\ gv'' \in B(\mathcal{D}), \quad |g(x)v''(x)| \leq C_2 x^{\alpha-1} (1-x)^{\alpha-1} \text{ on } (0,1); \\ gf(\cdot, v) \in B(\mathcal{D}), \quad |g(x)f(x, v(x))| \leq C_3 x^{\alpha-1} (1-x)^{\alpha-1} \text{ on } (0,1); \end{cases}$$

where

$$(4.4) \quad g(x) = x(1-x).$$

We shall also assume that the solution of the Frechet derivative problem

$$(4.5) \quad \theta''(x) + f_u(x, u(x))\theta(x) = w(x), \quad \theta(0) = \theta(1) = 0$$

satisfies

$$(4.6) \quad |\theta(x)| \leq C_4 \|A^{-1}w\|$$

for all $u \in M(d, \alpha)$ such that $\|u - u_0\| \leq \varepsilon$ where $\|\cdot\|$ is defined by

$$(4.7) \quad \|f\| = \sup_{x \in (0,1)} |f(x)|,$$

where

$$(4.8) \quad (A^{-1}f)(x) = -\int_0^1 G(x,t)f(t)dt$$

and where for any $x \in [0,1]$,

$$(4.9) \quad G(x,t) = \begin{cases} (1-x)t & \text{if } 0 \leq t \leq x \\ x(1-t) & \text{if } x \leq t \leq 1. \end{cases}$$

Moreover, we shall assume that if $\|u - u_0\| \leq \varepsilon$, then

$$(4.10) \quad \| \{A^{-1}f(t, u(t))\} \| \leq C_5.$$

Let us assume that we have found an approximate solution

$$(4.11) \quad u_m(x) = \sum_{k=-N}^N u_k S(k,h) \phi(x) \quad (m = 2N + 1)$$

by the method of the previous sections, and let us set

$$(4.12) \quad \theta_m = u_m - u_0$$

Then

$$(4.13) \quad \theta_m''(x) + f_u(x, \bar{u}(x)) \theta_m(x) = u_m''(x) + f(x, u_m(x))$$

for some \bar{u} between u_0 and u_m , and therefore, by (4.5) and (4.6),

$$(4.14) \quad |\theta_m(x)| \leq C_4 \|u_m + A^{-1}f(\cdot, u_m)\|.$$

Now by Theorem 2.10, we find, by taking $S_k(x) = g(x)S(k, h)\phi(x)$, $x_k = \frac{1}{2} + \frac{1}{2} \tanh(kh/2)$, that

$$(4.15) \quad \int_0^1 [v''(x) + f(x, v(x))] S_k(x) dx \approx h \frac{g(x_k)}{\phi'(x_k)} [v''(x_k) + f(x_k, v(x_k))]$$

and

$$(4.16) \quad \int_0^1 v''(x) S_k(x) dx \\ \approx h \sum_{j=-N}^N v(x_j) \left[\frac{g''(x_j)}{\phi'(x_j)} \delta_{kj}^{(0)} + \{2g'(x_j) + g(x_j)\phi''(x_j)/\phi'(x_j)\} \frac{\delta_{kj}^{(1)}}{h} \right. \\ \left. + g(x_j)\phi'(x_j) \frac{\delta_{kj}^{(2)}}{h^2} \right]$$

in which the error of either term on the right-hand side of (4.15) is bounded by $C_6 N^{-\frac{1}{2}} e^{-(\pi \alpha N)^{\frac{1}{2}}}$ and the error of the right-hand side of (4.16) is bounded by $C_7 N^{\frac{1}{2}} e^{-(\pi \alpha N)^{\frac{1}{2}}}$. By our process of solution, the numbers u_k in (4.11) are determined such that

$$(4.17) \quad h \sum_{j=-N}^N u_j \left[\frac{g''}{\phi'} \delta_{kj}^{(0)} + \{2g'_j + g_j \phi''/\phi'_j\} \frac{\delta_{jk}^{(1)}}{h} + g_j \phi'_j \frac{\delta_{kj}^{(2)}}{h^2} \right] \\ + h \frac{g_k}{\phi'_k} f(x_k, u_k) = 0, \quad k = -N, -N+1, \dots, N.$$

Theorem 4.2: Let the numbers u_k ($k = -N, -N+1, \dots, N$) be determined by (4.17), and let $u_m(x)$ be defined as in (4.11). Then

$$(4.18) \quad |u_m(x) - u_0(x)| \leq C_{15} N^{3/2} e^{-(\pi d \alpha N)^{1/2}}, \quad 0 \leq x \leq 1.$$

where u_0 is the solution of (4.1).

Proof: In view of the errors in the approximations (4.15) and (4.16), the solution of (4.17) is equivalent to finding a function $v \in M(d, \alpha)$, such that

$$(4.19) \quad \frac{g(x_k)}{\phi'(x_k)} [v''(x_k) + f(x_k, v(x_k))] = \frac{\epsilon_k}{h}, \quad k = -N, -N+1, \dots, N,$$

where $v(x_k) = u_k$, and where

$$(4.20) \quad |\epsilon_k| \leq C_8 N^{1/2} e^{-(\pi d \alpha N)^{1/2}}, \quad k = -N, -N+1, \dots, N.$$

Since $v \in M(d, \alpha)$, it follows, for any $t \in (0, 1)$, that

$$(4.21) \quad \begin{aligned} & \frac{g(t)}{\phi'(t)} [v''(t) + f(t, v(t))] - \sum_{k=-\infty}^{\infty} \frac{g(x_k)}{\phi'(x_k)} [v''(x_k) + f(x_k, v(x_k))] S(k, h) \phi(t) \\ &= \frac{\sin[\pi \phi(t)/h]}{2\pi i} \int_{\partial D} \frac{g(z) [v''(z) + f(z, v(z))] dz}{[\phi(z) - \phi(t)] \sin[\pi \phi(z)/h]} \end{aligned}$$

By multiplying (4.21) by $\phi'(t)^2$, taking A^{-1} of each side, and noting that $g(t)\phi'(t) = 1$, we get

$$\begin{aligned} & v(x) + \{A^{-1} f(t, v(t))\}(x) - \sum_{k=-\infty}^{\infty} \frac{g_k}{\phi_k} [v_k'' + f(x_k, v_k)] A^{-1} \{\phi'(t)^2 S(k, h) \phi(t)\}(x) \\ &= A^{-1} \left\{ \frac{\phi'(t)^2 \sin[\pi \phi(t)/h]}{2\pi i} \int_{\partial D} \frac{g(z) [v''(z) + f(z, v(z))] dz}{[\phi(z) - \phi(t)] \sin[\pi \phi(z)/h]} \right\}(x) \end{aligned}$$

Since $\phi'(t) = 1/[t(1-t)]$, it follows, by taking
 $t = [1 + \tanh(u/2)]/2$, $x = [1 + \tanh(w/2)]/2$, and using (4.8) and
(4.9), that

$$\begin{aligned}
 I_1(h, x) &\equiv A^{-1} \{ \phi'(t)^2 \operatorname{sinc}[\{\phi(t) - kh\}/h] \} (x) \\
 &= - \int_{-\infty}^w \frac{1 - \tanh(w/2)}{1 - \tanh(u/2)} \operatorname{sinc}\left[\frac{u - kh}{h}\right] du \\
 &\quad - \int_w^{\infty} \frac{1 + \tanh(w/2)}{1 + \tanh(u/2)} \operatorname{sinc}\left[\frac{u - kh}{h}\right] du
 \end{aligned}
 \tag{4.23}$$

On the interval $[-\infty, w]$, the function $[1 - \tanh(w/2)]/[1 - \tanh(u/2)]$ increases monotonically from $[1 - \tanh(w/2)]/2$ to 1 while on $[w, \infty]$, the function $[1 + \tanh(w/2)]/[1 + \tanh(u/2)]$ decreases monotonically from 1 to $[1 + \tanh(w/2)]/2$. For this reason, it may be shown by a somewhat lengthy, but simple argument, that

$$|I_1(h, x)| \leq 4\pi h. \tag{4.24}$$

Similarly if $x \in [0, 1]$ and $z \in \partial\mathcal{D}$, we can show that

$$|I_2(h, x)| \equiv \left| A^{-1} \left\{ \frac{\phi'(t)^2 \sin[\pi\phi(t)/h]}{2\pi i [\phi(z) - \phi(t)]} \right\} (x) \right| \leq \frac{2h}{d} \tag{4.25}$$

since $\operatorname{Im} \phi(z) = \pm d$.

By means of (4.19), (4.22) and (4.25), Eq. (4.21) now yields

$$\begin{aligned}
 &|v(x) + \{A^{-1} f(t, v(t))\}(x)| \\
 &\leq |I_1(h, x)| \sum_{k=-N}^N \frac{|\epsilon_k|}{h} + |I_1(h, x)| \sum_{|k| > N} \frac{g_k}{\phi_k} |v_k'' + f(x_k, v_k)| \\
 &+ |I_2(h, x)| \int_{\partial\mathcal{D}} \left| \frac{g(z)[v''(z) + f(z, v(z))]}{\sin[\pi\phi(z)/h]} \right| dz
 \end{aligned}
 \tag{4.26}$$

Using the bounds given in (4.20) and (4.24), we bound the first sum on the right-hand side of (4.26) by $C_9 N^{3/2} e^{-(\pi d \alpha N)^{1/2}}$; using (4.3) and (4.24), and recalling that $x_k = \frac{1}{2} + \frac{1}{2} \tanh(kh/2)$, we bound the second sum on the right-hand side of (4.26) by $C_{10} e^{-(\pi d \alpha N)^{1/2}}$; and using (4.25) and the fact that $|\sin[\pi \phi(z)/h]| \geq \sinh[\pi d/h]$ if $z \in \partial D$, we bound the integral term on the right-hand side of (4.24) by $2h N(g[v'' + f(\cdot, v)]) / [d \sinh(\pi d/h)] = C_{11} N^{-1/2} e^{-(\pi d \alpha N)^{1/2}}$. Hence for all $x \in [0, 1]$,

$$(4.27) \quad |v(x) + \{A^{-1} f(t, v(t))\}(x)| \leq C_{12} N^{3/2} e^{-(\pi d \alpha N)^{1/2}}.$$

Since $v \in M(d, \alpha)$, it follows from the first and second of (4.3) that

$$(4.28) \quad |v(x)| \leq C_{13} x^\alpha (1-x)^\alpha, \quad 0 \leq x \leq 1.$$

Furthermore, since $v \in M(d, \alpha)$, and since u_m and v coincide at $x_{-N}, x_{-N+1}, \dots, x_N$, it follows that [12, Theorem 8.2] for all $x \in [0, 1]$,

$$(4.29) \quad |u_m(x) - v(x)| \leq C_{14} N^{1/2} e^{-(\pi d \alpha N)^{1/2}}.$$

In view of (4.5), (4.6) and (4.10), it now follows that for all $x \in [0, 1]$,

$$(4.30) \quad \begin{aligned} & |u_m(x) - \{A^{-1} f(t, u_m(t))\}(x)| \\ & \leq |v(x) + \{A^{-1} f(t, v(t))\}(x)| + C_5 |u_m(x) - v(x)| \end{aligned}$$

By (4.14), (4.27), (4.29) and (4.30), it thus follows that for all $x \in [0, 1]$

$$(4.29) \quad |\theta_m(x)| = |u_m(x) - u_0(x)| \leq C_{15} N^{3/2} e^{-(\pi d \alpha N)^{1/2}}.$$

This completes the proof of Theorem 4.2.

Similarly, it may be shown that when using $n = (2N+1)^2$ points to obtain an approximate solution of a partial differential equation, such as (3.15), the error is bounded by $C_{16} N^{3/2} e^{-\gamma N^{1/2}} \leq 5C_{16} n^{3/4} e^{-\gamma n^{1/2}}$. Indeed, for the case of (3.15), we may take $C_{16} = 1$ and $\gamma = \pi^2$.

References

1. J. Chauvette and F. Stenger, The approximate solution of the non-linear equation $\Delta u = u - u^3$, J. Math. Anal. Appl. 51 (1975) 229-242.
2. Y.H. Chiu, An integral equation method of solution of $\Delta u = k^2 u$, Ph.D. thesis, University of Utah (1976).
3. G. Galub and W. Kahan, Calculating the singular values and Pseudo-inverse of a matrix, SIAM J. Numer. Anal. 2 (1965) 205-224.
4. R.V.L. Hartley, The Transmission of Information, Bell System Tech. J. 7 (1928) 535-560.
5. L. Lundin, A cardinal function method of solution of $\Delta u = u - u^3$, Ph.D. thesis, University of Utah (1975).
6. L. Lundin and F. Stenger, Cardinal-type approximations of a function and its derivatives, submitted for publication.
7. J. McNamee, F. Stenger and E.L. Whitney, Whittaker's cardinal function in retrospect, Math. Comp. 25 (1971) 141-154.
8. H. Nyquist, Certain topics in telegraph transmission theory, Trans. Amer. Inst. Elec. Engrg. 47 (1928) 617-644.
9. W. Petrick, J. Schwing and F. Stenger, An algorithm for the electromagnetic scattering due to an axially symmetric body with an impedance boundary condition, submitted for publication.
10. J. Schwing, Eigensolutions of potential theory problems in R^3 , Ph.D. thesis, University of Utah (1976).
11. C.E. Shannon, A mathematical theory of communication, Bell System Tech. J. 27 (1948) 379-423, 623-656.

12. Stenger, F., Approximations via Whittaker's cardinal function, J. Approx. Theory 17 (1976) 222-240.
13. Stenger, F., Kronecker product extension of linear operators, SIAM J. Numer. Anal. 5 (1968) 422-435.
14. E.T. Whittaker, On the functions which are represented by the expansions of the interpolation theory, Proc. Roy. Soc. Edinburgh 35 (1915) 181-194.
15. J.M. Whittaker, Interpolatory function theory, Cambridge, London (1935).

ON BOUNDARY EXTRAPOLATION AND DISSIPATIVE SCHEMES FOR HYPERBOLIC PROBLEMS

Moshe Goldberg*
 Department of Mathematics
 University of California
 Los Angeles, California, 90024

ABSTRACT. In this note we consider dissipative, stable approximations to well-posed linear hyperbolic initial value problems in the quarter plane $x \geq 0, t \geq 0$. We show that if boundary values are determined by extrapolation, then stability is maintained. This result was first suggested by Kreiss, and proved explicitly by the author. The proof is reviewed here using a stability criterion for a certain family of boundary conditions due to Goldberg and Tadmor.

The Lax-Wendroff scheme and other dissipative approximations are applied to a test problem. As expected from Gustafsson's rate-of-convergence theory, computations verify that if the boundary extrapolation and the difference scheme have equal order of accuracy, then this order is preserved.

1. INTRODUCTION. Consider the conservation law

$$(1a) \quad \partial u / \partial t + \partial g(u) / \partial x = 0, \quad x \geq 0, \quad t \geq 0,$$

and assume that the associated initial value problem

$$(1b) \quad u(x, 0) = f(x)$$

is well-posed in $L^2(0, \infty)$, so no boundary values are required at $x = 0$. This assumption implies that characteristic lines do not carry information from the exterior of the domain $x \geq 0, t \geq 0$ inward.

To approximate the initial value problem (1), we introduce a mesh size $\Delta x > 0, \Delta t > 0$; a grid function $v_v(t) = u(v\Delta x, t)$, $v = 0, \pm 1, \pm 2, \dots$; and a consistent, explicit finite difference scheme

$$(2) \quad v_v(t + \Delta t) = S\{v_{v-r}(t), \dots, v_{v+p}(t)\}; \quad v = 1, 2, 3, \dots,$$

r, p being fixed integers.

* This research was sponsored in part by the Air Force Office of Scientific Research, Air Force System Command, USAF, under Grant No. AFOSR-76-3046.

Since nonlinearity in (1a) leads to nonlinear dependence of $v_\nu(t + \Delta t)$ on the components of $v_{\nu-r}(t), \dots, v_{\nu+p}(t)$, we are unable to be more specific, at this stage, about the structure of the scheme S . However, we assume that S is L^2 -stable, in case it is applied to the pure initial value problem for $-\infty < x < \infty$.

Usually, $r > 0$, so it is impossible to approximate (1) by (2) without specifying boundary values at r grid points in some left neighborhood of the boundary $x = 0$. Thus, we admit boundary conditions of the form

$$(3) \quad v_\mu(t) = \sum_{j=1}^s c_j v_{\mu+j}(t), \quad \mu = 0, \dots, -r + 1,$$

where the coefficients c_j and $s \geq 1$ are fixed. That is, having the values $v_\nu(t)$, $\nu \geq 1$, computed by the basic scheme (2), we proceed, at each time step, by using (3) to determine $v_\mu(t)$, $\mu = 0, -1, \dots, -r + 1$, in that order.

A natural way to choose the boundary conditions in (3) would be to employ extrapolation of degree $s - 1$ -- a procedure which is of accuracy of order s . More explicitly, we extrapolate from $v_1(t), \dots, v_s(t)$ to $v_0(t)$; then from $v_0(t), \dots, v_{s-1}(t)$ to $v_{-1}(t)$, etc. With the use of Stirling's extrapolation formula, (3) becomes

$$(4) \quad v_\mu(t) = \sum_{j=1}^s \binom{s}{j} (-1)^{j+1} v_{\mu+j}(t), \quad \mu = 0, \dots, -r + 1.$$

The main purpose of this note is to study the influence of boundary extrapolation on the stability of the numerical algorithm. This question is discussed in Section 2, where we consider a scalar linear conservation law which we approximate by a dissipative scheme. In this simple case it is shown that boundary extrapolation maintains stability. This result, which was first suggested by Kreiss, [5], was proven explicitly in [1], using Kreiss' theory, [6], for dissipative approximations of mixed initial-boundary value problems.

In fact, the above assertion is an immediate corollary of a forthcoming work by Goldberg and Tadmor, [2], which provides stability criteria for some general families of boundary conditions, including those presented in (3).

Finally, in Section 3, the Lax-Wendroff scheme, [7], and a new 5-point dissipative approximation by Gottlieb and Turkel, [3], are applied to a test problem. The numerical results support Gustafsson's rate-of-convergence theory, [4], by showing that the accuracy order of the basic scheme is maintained, if the extrapolation at the boundary is of the same order. The computation were carried out at the Campus Computing Network of the University of California at Los Angeles.

2. STABILITY ANALYSIS. From now on we restrict attention to the linear, scalar version of (1), namely to the initial value problem

$$(5) \quad \partial u / \partial t + a \partial u / \partial x = 0; \quad a = \text{const.}; \quad x \geq 0, \quad t \geq 0; \quad u(x, 0) = f(x),$$

which is well-posed if and only if $a < 0$.

Our explicit approximation in (2) becomes

$$(6) \quad \begin{aligned} v_v(t + \Delta t) &= Q v_v(t), \quad v = 1, 2, 3, \dots, \\ Q &= \sum_{j=-r}^p a_j E^j, \quad E v_v = v_{v+1}, \quad r > 0, \end{aligned}$$

where the constants a_j depend on a and on the fixed ratio $\lambda = \Delta t / \Delta x$, and initial values are determined by

$$v_v(0) = f_v, \quad v = 1, 2, 3, \dots$$

The assumption of dissipativity is that for some $\delta > 0$ and natural ω , the amplification factor of the scheme,

$$\hat{Q}(\xi) = \sum_{j=-r}^p a_j e^{ij\xi}, \quad -\pi \leq \xi < \pi,$$

satisfies

$$|\hat{Q}(\xi)| \leq 1 - \delta |\xi|^{2\omega}, \quad \forall |\xi| \leq \pi.$$

Thus, it is evident that $\hat{Q}(\xi)$ is now power bounded (by 1), which is well known to be equivalent to the (strong) stability of our basic scheme.

Introducing the boundary conditions (4), the concept of stability

becomes considerably more complicated, and we review it briefly. Let $H \equiv H(\Delta x)$ be the space of all grid functions, $w = \{w_v\}_{v=-r+1}^{\infty}$, which satisfy $\sum_{v=-r+1}^{\infty} |w_v|^2 < \infty$ and fulfill the boundary conditions in (4). If inner product and norm are defined by

$$(v, w) = \Delta x \sum_{v=-r+1}^{\infty} v_v \bar{w}_v, \quad \|w\|^2 = (w, w),$$

then H becomes a discrete analogue of $L^2(0, x)$.

Having constructed H , we realize that our finite difference algorithm in (4) and (6) defines a linear, bounded operator, $G : H \rightarrow H$, such that the numerical solution v satisfies,

$$v(t + \Delta t) = Gv(t), \quad \text{for } v(t) \in H.$$

Since

$$v(t) = G^m v(0) \quad \text{for } t = m\Delta t, \quad m = 1, 2, 3, \dots,$$

stability means that the powers of G are uniformly bounded, i.e., that for some constant K ,

$$\|G^m\| \leq K, \quad m = 1, 2, 3, \dots$$

We are now ready to state the main result:

THEOREM 1. Let the initial value problem (5) be approximated by an arbitrary, dissipative (stable) scheme of type (6), which is complemented by boundary extrapolation of arbitrary order. Then, the overall numerical algorithm is stable.

The proof which is laid out in [1], is a direct but somewhat lengthy application of Kreiss' stability theory, [6], for dissipative schemes. As required by Kreiss' criterion, the problem was to show that the corresponding operator G has no eigenvalues z in the unit disk.

In a forthcoming paper, [2], Goldberg and Tadmor use Kreiss' theory to provide a particularly simple stability condition in the case where scheme (6) is augmented by boundary conditions of type (2). This condition is rephrased as follows:

THEOREM 2 (Goldberg, Tadmor). Let (6) be an arbitrary, dissipative (stable) approximation, augmented by boundary conditions of type (2), then the overall algorithm is stable if

$$\sum_{j=-r}^p c_j \kappa^j \neq 1, \quad \forall \kappa \quad \text{with} \quad |\kappa| < 1.$$

Theorem 2, which is actually independent of the basic scheme, yields Theorem 1 immediately. For, considering the boundary conditions in (4), we want to show that

$$\sum_{j=1}^s \binom{s}{j} (-1)^{j+1} \kappa^j \neq 1 \quad \text{for} \quad \kappa \quad \text{with} \quad |\kappa| < 1,$$

i.e., that for all κ with $|\kappa| < 1$,

$$(1 - \kappa)^s \equiv \sum_{j=0}^s \binom{s}{j} (-\kappa)^j \neq 0.$$

The last inequality holds, and Theorem 1 follows.

3. NUMERICAL RESULTS. Consider the test problem

$$(7) \quad \partial u / \partial t - \partial u / \partial x = 0; \quad x \geq 0, \quad t \geq 0; \quad u(x, 0) = \sin 2\pi x$$

whose analytic solution is

$$u(x, t) = \sin 2\pi(x + t).$$

The second order accurate Lax-Wendroff scheme (L-W), [7], is in this case

$$(8) \quad v_v(t + \Delta t) = \frac{\lambda}{2}(\lambda - 1)v_{v-1}(t) + (1 - \lambda^2)v_v(t) + \frac{\lambda}{2}(\lambda + 1)v_{v+1}(t), \quad \lambda = \frac{\Delta t}{\Delta x},$$

and it is well known (e.g., [8, Chapter 12]) that dissipativity, and hence stability, are guaranteed if $\lambda < 1$.

In order to apply (8) to (7) we need to specify only one boundary value, $v_0(t)$, which according to (4), is given by

$$v_0(t) = \sum_{j=1}^s \binom{s}{j} (-1)^{j+1} v_j(t).$$

Here the accuracy of the boundary extrapolation is of orders s where s is arbitrary.

In Table 1 we compare the L-W results with the analytic solution at $t = 1$. The H-norm of the error was computed over the interval $0 \leq x \leq 1$, and is defined by

$$(9) \quad \|e\|_{(0,1)}^2 = \Delta x \sum_{v=0}^J [v_v(1) - u(v\Delta x, 1)]^2, \quad J = 1/\Delta x.$$

Δx	s	m	$\ e\ _{(0,1)}$
.05	2	40	5.57 - 2
.025	2	80	1.39 - 2
.05	1	40	7.03 - 2
.025	1	80	2.23 - 2

Table 1. L-W results at $t = 1$;
 $\lambda = 1/2$; $m = t/\Delta t$ is number of time steps.

Gustafsson, in his rate-of-convergence theory, [4], has discussed situations similar to the one under consideration. He has shown that in order to maintain the accuracy of the basic scheme, it is sufficient to employ boundary conditions of the same order of accuracy. Indeed, Table 1 suggests that L-W's second order accuracy is maintained if the boundary extrapolation is linear ($s = 2$), but is reduced if $s = 1$.

A second example is concerned with a family of centered, 5-point, dissipative schemes by Gottlieb and Turkel (G-T), [3]. The family, given in (2.4) of [3], depends on two parameters α and σ . Choosing $\alpha = 1/2$, $\sigma = 1$, and linearizing, we obtain an approximation to (7) of the form

$$\begin{aligned} v_v(t + \Delta t) = & -\frac{\lambda}{4}\left(\frac{\lambda}{2} - \frac{1}{3}\right)v_{v-2}(t) + \lambda\left(\lambda - \frac{2}{3}\right)v_{v-1}(t) + \left(1 - \frac{7}{4}\lambda^2\right)v_v(t) \\ & + \lambda\left(\lambda + \frac{2}{3}\right)v_{v+1}(t) - \frac{\lambda}{4}\left(\frac{\lambda}{2} + \frac{1}{3}\right)v_{v+2}(t), \quad \lambda = \frac{\Delta t}{\Delta x}, \end{aligned}$$

where the dissipativity condition is $\lambda < \sqrt{2}/2$.

Now we need two boundary values which are given by

$$v_\mu(t) = \sum_{j=1}^s \binom{s}{j} (-1)^{j+1} v_{\mu+j}(t), \quad \mu = 0, -1.$$

The error-norms in Table 2 are computed, as in the previous case.

over $0 \leq x \leq 1$, and in analogy to (8) are defined by

$$\|e\|_{(0,1)}^2 = \Delta x \sum_{v=-1}^J [v_v(1) - u(v\Delta x, 1)]^2, \quad J = 1/\Delta x.$$

Δx	λ	s	m	$\ e\ _{(0,1)}$
.05	.5	4	40	1.85 - 2
.025	.25	4	160	1.16 - 3
.05	.5	3	40	2.42 - 2
.025	.25	3	160	1.92 - 3

Table 2. G-T results for $t = 1$.

Unlike the L-W scheme which is of second order accuracy both in time and space, the G-T approximation is of second order in time and fourth order in space. Since the boundary extrapolation is taken only with respect to the space variable, it should be expected that in order to maintain the fourth-order accuracy in x , we have to utilize cubic extrapolation ($s = 4$), regardless of the fact that G-T's accuracy in time is only of second order. This is reflected by the results of Table 2.

REFERENCES

1. M. Goldberg, On a boundary extrapolation theorem by Kreiss, Math. Comp., Vol. 31 (1977), 469-477.
2. M. Goldberg and E. Tadmor, to appear.
3. D. Gottlieb and E. Turkel, Dissipative two-four methods for time dependent problems, Report 75-22, ICASE, NASA Langley Research Center, Hampton, Virginia.
4. B. Gustafsson, The convergence rate for difference approximations to mixed initial boundary value problems, Math. Comp., Vol. 29 (1975), 396-406.
5. H. O. Kreiss, Difference approximations for hyperbolic differential equations, Numerical Solution of Partial Differential Equations, (Proc. Sympos., Univ. of Maryland, College Park, Maryland 1965), Academic Press, New York, 1966, pp. 51-58.

6. H. O. Kreiss, Stability theory for difference approximations of mixed initial boundary value problems. I, Math. Comp., Vol. 22 (1968), 703-714. MR 39 # 2355.
7. P. D. Lax and B. Wendroff, Systems of conservation laws, Comm. Pure Appl. Math., Vol. 13 (1960), 217-237. MR 22 # 11523.
8. R. D. Richtmyer and K. W. Morton, Difference Methods for Initial Value Problems, 2nd ed., Interscience Tracts in Pure and Appl. Math., Vol. 4, Wiley, New York, 1967. MR 36 # 3515.

ELLPACK: A COOPERATIVE EFFORT FOR THE STUDY OF NUMERICAL METHODS
FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

John R. Rice
Purdue University

1. BACKGROUND. In the summer of 1975, Garrett Birkhoff started discussing the possibility of a cooperative effort to develop and evaluate software for elliptic partial differential equations. In the summer of 1976, James Ortega organized a small meeting of interested parties to explore various viewpoints and possibilities for this idea. A framework was outlined which seemed to accommodate a number of people's work and interests, two initial projects were agreed upon and John R. Rice was selected as coordinator of the ELLPACK effort.

Participation in ELLPACK is completely voluntary. Purdue University will provide the software framework and define the structure precisely. Contributors can prepare programs which fit into this framework and Purdue will incorporate them into ELLPACK. It is assumed that contributors will submit high quality, portable programs to ease the burden of integrating programs into ELLPACK.

2. ELLPACK OBJECTIVES. The primary objective of ELLPACK is as a tool for research in the evaluation and development of numerical methods for solving elliptic partial differential equations. Various software components can be interchanged and the resulting performance (accuracies, efficiencies, etc.) can be measured.

ELLPACK's use as a research tool requires its framework to be convenient, flexible and modular. Thus it will be suitable for educational use and for others who have easy or moderately difficult problems. It is not intended that ELLPACK be directly applicable to the very complex applications (e.g. temperature distribution in a nuclear power plant or in a reentry vehicle). Nevertheless, the ability to quickly solve a variety of moderately complex problems should be valuable in many areas.

3. THE TWO PROJECTS. The first project is ELLPACK 77 which is based on adaptations of existing programs. One of its primary objectives is to test the concept of a modular approach using interchangeable software parts. ELLPACK 77 is restricted to rectangular geometry in 2 or 3 dimensions. Anticipated capabilities include:

Operator Approximation

2-Dimensions:	5-point star, 9-point star, Collocation and Galerkin with Hermite cubics
3-Dimensions:	7-point star, 27-point star
Special Options:	Poisson Problem, Constant Coefficients, Self Adjoint Form.

Equation Solution

Direct Elimination (Band or Profile)
Nested Dissection
"Fast" Methods (Tensor Product and FFT)
SOR, Accelerated Jacobi, etc.
Conjugate Gradient

The second project is ELLPACK 78 where the primary extension is to non-rectangular geometry, an area where some group members are already active. Other directions which may be followed include:

- a. Standard, automatic changes of variables
 - b. Enhancement of rectangular domain capabilities
 - c. More operator approximations, e.g.
 HODIE methods, Hermite cubics in 3 dimensions, Method of particular solutions, Capacitance methods
 - d. More equation solvers, e.g.
 Cyclic Chebyshev, Automated selection of SOR parameters
3. Parallel processor implementation

It is hoped that a significant part of these capabilities will be implemented by late 1978. At that point the ELLPACK effort will be evaluated and future efforts, if any, considered.

4. TECHNICAL OPERATION. The framework and ELLPACK specifications are specified at Purdue University. Careful attention is given to making ELLPACK compatible with a wide range of interests and to making it "easy" to contribute to ELLPACK. On the other hand, success depends on certain uniform standards and conventions and there is no doubt that choices will be made which some contributors find inconvenient. It is assumed that contributors are experienced in producing portable, understandable and good quality software. The main technical document is the Contributor's Guide to ELLPACK [Rice, 1976] which defines the ELLPACK environment for a potential contributor. There is also a shorter User's Guide [Rice, 1977] and a guide to implementing ELLPACK at locations other than Purdue.

The following assumptions are made about the environment for ELLPACK:

- A. A standard Fortran compiler is available.
- B. The operating system allows Fortran preprocessors.
- C. The operating system allows a reasonable number of files (probably on disks) to be created by a job and simply manipulated. Libraries of compiled programs are possible.
- D. Certain common utility routines are available. These include file copying and concatenation, timing routines, and hard copy graphical output. The lack of some of these utilities can be circumvented by deleting the corresponding features from ELLPACK.

5. ELLPACK 77 PREPROCESSOR. The user defines a problem to be solved (including a fixed grid to be used) and then specifies a combination of modules to be used in its solution. The basic elements in ELLPACK are segments and we list the segment types along with a brief indication of their use. An example ELLPACK 77 program is given at the end of this paper.

Group 1: Must appear (only once) and be before any from Group 2.

EQUATION. Specifies the partial differential equation.

BOUNDARY. Specifies the domain and the boundary conditions to be satisfied.

GRID. Specifies the rectangular grid in the domain.

Group 2: May be used as needed and repeated if desired.

DISCRETIZATION. Specifies the method to discretize the equations and to generate the linear equations.

INDEXING. Places the equations generated by the discretization module in some desired order for further processing.

SOLUTION. Actually solves a linear system of equations.

OUTPUT. Provides various requested information.

Group 3: May appear anywhere and as often as desired.

* A comment.

OPTIONS. Specifies various options selected.

FORTRAN. Contains Fortran subprograms used by other segments.

Group 4: Must appear after all Group 2 segments.

SEQUENCE. Specifies the sequence for the Group 2 segments to be used.

Every ELLPACK program ends with the segment END.

Most of the features of ELLPACK 77 input are illustrated in the example below. Note that ELLPACK 77 is heavily key-word oriented. Standard naming conventions are used such as referring to the solution as U and its various derivatives as UX, UYY, etc. The coordinates are always called X, Y and Z. See [Rice, 1977] for the detailed syntax of the ELLPACK segments. See [Rice, 1977a] for further discussion of ELLPACK.

This work is supported in part by a grant from the National Science Foundation.


```

*          ELLPACK77 EXAMPLE - MARCH 18, 1977
*
* OPTIONS.          DEBUG = 2
*
* EQUATION.      2 DIMENSIONS
                  UXX$ +6, UYY$ -X**2 UX$ + 1./(X+DUB9(Y)) U = COS(Y*X)
*
* BOUND.      X = 0.0      , U = 0.0
               Y = 1.0      , UY = X
               X = EXP(1.) , U = EXP(Y)
               Y = 0.0      , MIXED = (1.2)U (COS(X+.2))UY = EXP(X)
*
* GRID.          UNIFORM X = 7
                  NGRIDY = 11, 0., 0.08, 0.16, 0.24, 0.35, 0.46, 0.575, 0.7, 0.8
                  125, 0.925, 1.0
*
* DISCRETIZATION. 5-POINT STAR
*
* INDEX(1).      NATURAL
* INDEX(2).      RED-BLACK
*
* SOL.           SOR1
*
* OUTPUT(AA).     PLOT-TRUE $ PLOT-DOMAIN
* OUT(B).         PLOT-ERROR $ MAX-ERROR
* OUTPUT(99).     TABLE(5,5) = U
*
* OPTIONS.        TIME $ MEMORY
*
* SEQUENCE. OUTPUT(AA) $ DIS $ INDEX(1) $ SOLUTION $ OUTPUT(B)
                  INDEX(2) $ SOLUTION $ OUTPUT(B)
                  OUTPUT(99)
*
* FORTRAN.
      FUNCTION TRUE(X,Y)
      TRUE = EXP(X+Y)/(1.+DUB9(Y))**2
      RETURN
      END
      FUNCTION DUB9(T)
      DUB9 = T*(T+.5)*EXP(T/(1.+T*COS(T)))
      RETURN
      END
END.

```

REFERENCES

1. John R. Rice, ELLPACK: A Cooperative Effort for the Study of Numerical Methods for Elliptic Partial Differential Equations, CSD-TR 203, Computer Science, Purdue University, October 15, 1976, 6 pages.
2. John R. Rice, ELLPACK Contributor's Guide, CDS-TR 208, Computer Science, Purdue University, November 1, 1976, 44 pages.
3. John R. Rice, ELLPACK 77 User's Guide, CDS-TR 226, Computer Science, Purdue University, March 18, 1977, 17 pages.
4. John R. Rice, ELLPACK, A Research Tool for Elliptic Partial Differential Equations Software, in "Mathematical Software - Madison 1977" (J. Rice, ed.), Academic Press, New York, 1977.

Mathematical Sciences
Purdue University
West Lafayette, Indiana 47907

STORAGE AND RETRIEVAL OF INFORMATION ON SYSTEMS OF
PARTIAL DIFFERENTIAL EQUATIONS AND THEIR SOLUTIONS:
CREATABASE AND THE CONTINUUM MECHANICS CENTER DATA
BASE OF HYDROCODES

Morton A. Hirschberg
Joseph Lacetera
James A. Schmitt

U. S. Army Ballistic Research Laboratory
Ballistic Modeling Division
Aberdeen Proving Ground, Maryland 21005

ABSTRACT

A Continuum Mechanics Center has been established for the purposes of evaluating and developing models of interacting continua. Because of the large and growing body of literature concerning such models and related computer codes, the vast number of assumptions made in their use, and the varying types of numerical methods utilized in these codes, a data base analysis system, CREATABASE,¹ was used to store information and characteristics of the different codes.

This paper briefly describes CREATABASE, delineates the data base, describes queries made on the data base and outlines future uses and expansion of the data base and the data base analysis system.

¹ Daniel Analytical Services Corporation, "User Reference Manual for The CREATABASE Module of an Integrated Data Base Analysis System: Level U-4A," Houston, TX, August 1976.

I. INTRODUCTION

The formation of a Continuum Mechanics Center (CMC) at the Ballistic Research Laboratory (BRL) to study, evaluate, and develop large hydrodynamics, solid mechanics, particle transport, and heat transfer computer codes presented an excellent opportunity to simultaneously generate a data base containing information on systems of partial differential equations and their solutions.

A questionnaire (Appendix I) was developed and sent to a number of BRL scientists soliciting information regarding codes of interest. The response furnished data on 20 codes and led to the formation of a data base from which significant information can be derived.

CREATABASE, a commercial data base analysis system marketed by Daniel Analytical Services Corporation (DANAYLT) of Houston, Texas, was used to store data for retrieval. CREATABASE is a relational ^{2,3} data base system, written in FORTRAN, which runs on the UNIVAC 1100 series computers.

Queries are made using English-like statements and may be made in a batch or interactive processing mode. The output for each mode is slightly different. CREATABASE affords little in the way of report generation; that is, formatted output. CREATABASE does, however, offer the user the capability of outputting all or any part of the data on an auxiliary file which the user can then process in any fashion desired including report formats.

Sixty-two descriptors form the total domain of the current data base. It offers the user an accessible and easily used tool for ascertaining characteristics and capabilities of certain computer codes at BRL. Information such as the code's applications, numerical method, spatial geometry, equation(s) of state and reports dealing with the code and its performance, as well as 32 other items, are included in the data base. However, specific data about solutions of equations such as subroutine names in which various processes occur, the actual equations solved, or anomalies of systems of equations in a particular code do not now constitute a part of the data base. Although this information should be available in a user's manual, a more accessible information source is desirable. As the data base develops such items will be considered as possibilities for inclusion.

² Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," Communications of the Association for Computing Machinery, 13, No. 6, June 1970.

³ Date, C.J., An Introduction to Data Base Systems, Addison-Wesley, NY, 1976.

Adding new information on codes already in the data base and cataloging other codes are a continuing part of the CMC's activities. In addition, new commands to allow easier querying are being added to the CREATABASE system.

II. THE CREATABASE SYSTEM

CREATABASE is a relational data base analysis system; that is, all of the data which forms the data base exists in tabular form. The columns are formally called descriptors (domains or sets in relational terms) and contain all of the states which comprise that descriptor. For example, a descriptor might be MAXIMUM SPATIAL DIMENSIONALITY and contain as states ONE DIMENSIONAL, TWO DIMENSIONAL, AND THREE DIMENSIONAL. Rows are formally called records (n-tuples or relations) and are formed by selecting one of the possible states from each descriptor. See Figure 1 as an example of an input record (note that two successive commas indicate there is no data for that entry).

CREATABASE is a compiler, written in FORTRAN, that takes statements written in an English-like language, interprets them and executes them. The program is very compact requiring only 21,000 words of storage, yet is very modular consisting of 42 subroutines.

There are 56 commands in CREATABASE which fall into seven command categories (see Figure 2). Since an explanation for each command appears elsewhere (see Reference 1) the commands will not be discussed in great detail here. It should be noted, however, that a data base can be created and queried with as few as four commands.

CREATABASE does not have an extensive report generation capability. It does indicate how many hits or matches have occurred and what percentage of the data base the number of hits represent. This statistical information can be used for designing further queries and to check the validity of the data base itself.

In addition, CREATABASE does allow any or all of the data in the data base to be output onto a file for further processing during that execution or at a later time. This selective retrieval of data for future use is a most useful tool for scientific processing. Several independent programs exist to assist the user in unformatting data for his special applications. The user then has great control over the subsequent handling of his data in addition to the capabilities provided by the system itself. The user may interface his data with graphics, simulation, statistical, or reports generation packages. The user may also interface CREATABASE with other data base management packages; for example, using CREATABASE for the purpose of collecting and refining data and the other packages for elegant output forms.

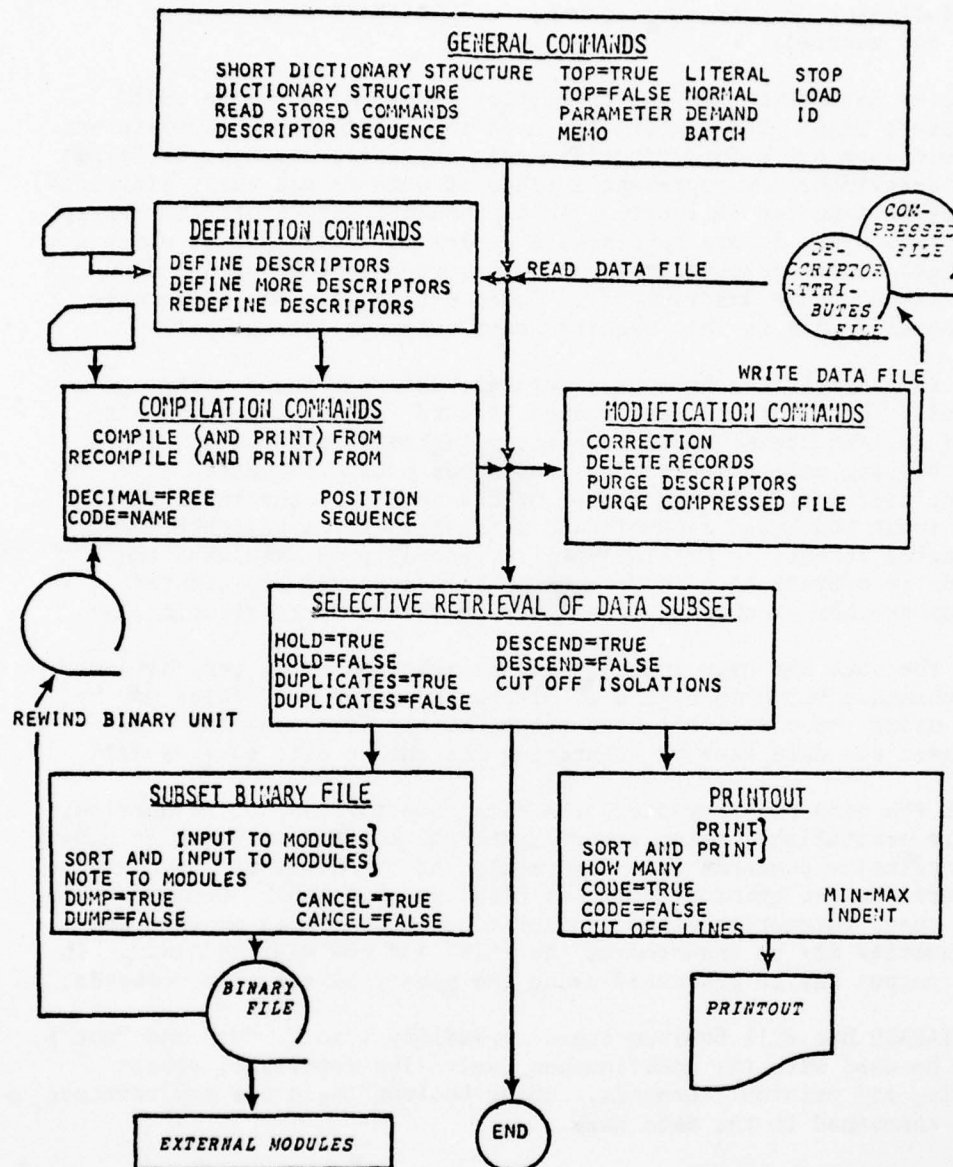


Figure 2. The CREATABASE Commands and Command Categories

The following describes briefly how a CREATABASE data base is assembled and queried.

The user determines his own descriptor names which may be up to 126 characters long. Descriptors are used to represent both numeric and alphanumeric data but a descriptor may only represent one type of data. A numeric descriptor may represent a range of numbers and carry additional identifying information (a label). An alphanumeric descriptor may be in either of two forms; it may represent a series of states all of which are predetermined (coded descriptors), or it may represent an open ended series of states (name descriptors). Coded descriptors are preferable for alphanumeric data as they require fewer computer storage cells.

Once the descriptors have been defined, data are input using one of three forms. The first and most common is card input. Input in this fashion is in free format and may be accomplished in a batch or interactive processing mode. The second input mode uses a formatted file whereby the user indicates the number of characters in the input string. The third input mode uses a CREATABASE file which is in a highly coded, densely packed format. A CREATABASE file generally is used when the user wishes to operate on a subfile which he has previously created (output) on another execution, or earlier in the current execution.

Once the data has been input it may be examined using general and printout commands shown in Figure 2. If there are errors, these may be corrected using the modification or subset binary file commands. One might correct the data base by reentering the entire data base as well.

After the data are corrected, the data base is ready to be queried. Queries are accomplished using general, selective retrieval, subset binary file, and printout commands (see Figure 2). At the heart of the querying commands are the two printout commands PRINT and HOW MANY. Commands from the other query categories allow manipulations of the data so that the specific queries may be answered by the PRINT and HOW MANY commands. In addition, output may be generated using the subset binary file commands.

CREATABASE has full Boolean logic capability ("and", "or" and "not") which can be used with the modification, selective retrieval, subset binary file, and printout commands. Using Boolean logic one may retrieve any datum contained in the data base.

The general form of the output commands is a descriptor list (the desired output) followed by a Boolean expression. A Boolean expression is a concatenation of quantifiers using the Boolean operators "and", "or" and "not". A quantifier has the form descriptor name, followed by a descriptor value (state).

Samples of CREATABASE queries and the resulting output are shown in Section V.

In addition to Reference 1, an annotated guide⁴ and two samples of CREATABASE runs⁵ are most helpful for using and understanding the CREATABASE commands and their interactions. The cited references, while forming a complete set of CREATABASE system documents, are terse and make the use of the system seem more complicated than it is. Of course, quite complex interactions can be obtained through the use of CREATABASE and the UNIVAC EXEC VIII operating system. Such interactions, while noted, will not be discussed here.

III. THE CONTINUUM MECHANICS CENTER DATA BASE

Data for the CMC data base was gathered using the questionnaire shown in Appendix I. Questionnaires were sent to a number of BRL scientists who supplied data which were then used to define the descriptors (state names) for the data base. Sixty-two descriptors (see Figure 3) were used to describe the data. The data base was designed so that each record of data provided information for one code. The descriptors are divided into several broad categories: those dealing with (i) the type of problems treated by the code; for example, descriptors 8, 9, 10, 15, 16, 17, 18, 24, 28, 29, 31, 32, 34, 36, 37, (ii) the characteristics of the code; for example, descriptors 1, 2, 5, 6, 7, 11, 12, 13, 14, 19, 20, 21, 22, 23, 25, 26, 27, 30, 33, 35, 38, 39, 40, and (iii) people and reports connected with the code; for example, descriptors, 3, 4, 41-62.

Queries are often initially made on certain descriptors to determine which code(s) can perform a desired type of calculation. Subsequent queries can then be made to obtain more detailed information concerning these codes. (For an example set of queries, see Section V). Furthermore, other data bases can be generated from the current data base; for example, if the data base becomes very large, one consisting only of reports dealing with the codes may become desirable.

The data base which is stored in 35,000 words on the UNIVAC 1108 computer currently contains data for 20 codes (see Appendix II). Although there are now only 20 records in the data base significant information

⁴ Daniel Analytical Services Corporation, ""Primer" for "The CREATABASE Module" of An Integrated Data Base Analysis System: Level U-4A," Houston, TX, August 1976.

⁵ Daniel Analytical Services Corporation, "An Illustrative Check Deck for "The CREATABASE Module" of An Integrated Data Base Analysis System: Level U-4A," Houston, TX, August 1976.

can be extracted (see Section V). Future plans include expanding the data base to include more codes and more reports. However, this data base will not become a bibliography for different hydrocodes.

IV. USING THE DATA BASE

The data base is operational on the UNIVAC 1108 computer at Edgewood. As such, it runs under the EXEC VIII operating system. This section will provide the user the means to sign onto the computer, invoke the CREATABASE system, and gain access to the data base. It is highly recommended that users copy the data base files onto their own files before using the system. If this is impractical, the user must not invoke any commands which would modify the data base; that is, the user must not use any of the definition, modification, or compilation commands.

The following describes BATCH mode operation. To sign onto the computer the command in card column 1 is:

```
@RUN IDENTIFICATION, ACCOUNT NUMBER, CMCLIB, TIME, PAGES OF OUTPUT.
```

The user must make arrangements for obtaining an account number. CMCLIB is the project name for the CMC CREATABASE data base. The next instruction (card column 1) is:

```
@MISD*CAB.CAB CMDIC1.D, CMCMP1.C.
```

The MISD*CAB.CAB invokes the CREATABASE system; CMDIC1.D is the file containing the descriptor attributes (logical unit 9; see Reference 1) and CMCMP1.C is the file containing the compressed data (logical unit 12).

At this time control passes from the UNIVAC EXEC VIII operating system to CREATABASE. You are ready to query the data base using any of the permissible command categories: general, selective retrieval, subset binary, or printout. A familiarity with the CREATABASE system is helpful to minimize the time spent in designing queries and auxiliary output (using the subset binary file operations). CREATABASE commands are free form; that is, there are no card column restrictions as to where commands can be placed. The normal CREATABASE separator is the comma and the normal command terminator is the asterisk. Not all CREATABASE commands need a terminator; however, the user is unburdened by using a terminator on all commands. The user has the option of changing the separator and terminator if he so desires.

1. CODE NAME
2. GENERAL DESCRIPTION
3. USERS MANUAL
4. USERS MANUAL AUTHOR
5. COMPUTER LANGUAGE
6. COMPUTERS ON WHICH CODE IS OPERATIONAL
7. DEVELOPMENTAL STATUS
8. PRIMARY APPLICATION
9. SECONDARY APPLICATION
10. TERTIARY APPLICATION
11. MESH TYPE
12. GENERAL NUMERICAL METHOD
13. PARTICULAR NUMERICAL METHOD
14. ORDER OF SCHEME
15. MAXIMUM SPATIAL DIMENSIONALITY
16. SPATIAL GEOMETRY
17. UNSTEADY CALCULATION
18. CONSERVATION/TRANSPORT EQUATIONS SOLVED
19. VARIABLES COMPUTED FROM CONSERVATION/TRANSPORT EQUATIONS
20. EQUATION OF STATE 1
21. EQUATION OF STATE 2
22. EQUATION OF STATE 3
23. EQUATION OF STATE 4
24. BOUNDARY CONDITIONS
25. MAXIMUM GRID SIZE
26. TYPE OF REZONING
27. EXPANDING GRID
28. MATERIAL RESPONSE
29. ELASTIC PLASTIC SOLID
30. NUMBER OF MATERIALS
31. INTERFACE CAPABILITY
32. TYPE OF FLUID FLOW
33. SHOCK TREATMENT
34. TYPE OF RADIATION TRANSPORT
35. TYPE OF ENERGY DEPOSITION
36. TYPE OF CHEMICAL REACTIONS
37. TYPE OF ATOMIC REACTIONS
38. EXTENT OF GRAPHICS CAPABILITY
39. SPECIAL FEATURES
40. LIMITATIONS
41. KNOWLEDGEABLE USERS
42. NUMBER OF REPORTS
43. REPORT TITLE 1
44. REPORT AUTHORS 1
45. REPORT TITLE 2
46. REPORT AUTHORS 2
47. REPORT TITLE 3
48. REPORT AUTHORS 3
49. REPORT TITLE 4
50. REPORT AUTHORS 4
51. REPORT TITLE 5
52. REPORT AUTHORS 5
53. REPORT TITLE 6
54. REPORT AUTHORS 6
55. REPORT TITLE 7
56. REPORT AUTHORS 7
57. REPORT TITLE 8
58. REPORT AUTHORS 8
59. REPORT TITLE 9
60. REPORT AUTHORS 9
61. REPORT TITLE 10
62. REPORT AUTHORS 10

Figure 3. The Sixty-Two Descriptors Used for the Continuum Mechanics Center Data Base

When one has finished his CREATABASE operations, control is given back to the UNIVAC system with the following command (card column 1):

@FIN

This command will provide time and cost information to the user.

If the user wishes to query CREATABASE using the interactive mode, several additional commands are necessary. First, the user must dial up and be given access to the computer. Next, before using the RUN statement, the user must identify himself using a site identification. Site identifications are easily obtained and are well marked on hard wired terminals. After the site ID has been entered and the computer has acknowledged it, the procedure is as described above. At the conclusion of the terminal execution, after the @FIN command has been issued, the user must issue an @@TERM and wait for the terminal or modem light to go out.

Additional aids for the user are the commands CNTRL Z to erase the last character typed if a mistake was made and @@X T10 to interrupt output when a query is producing too much output. Greater knowledge of the EXEC VIII operating system and CREATABASE only enhances the skill of the user and enables him to do more complication operations. However, the information presented here is sufficient to query the data base.

V. SAMPLE QUERIES AND OUTPUTS

This section will show several typical queries and the instructions used prior to the queries so that the user has the proper information for querying at his disposal. A complete list of the 62 descriptors in the CMC data base can be obtained by using the following command:

```
>SHORT DICTIONARY STRUCTURE*  
SHORT DICTIONARY STRUCTURE*
```

Notice that the command is echoed back to the user which accounts for the repeated line of output. The output of this command is given in Figure 3. The individual states of any descriptor can easily be determined; for example, the states of the descriptors GENERAL NUMERICAL METHOD, MAXIMUM SPATIAL DIMENSIONALITY, SPATIAL GEOMETRY and TYPE OF FLUID FLOW, are obtained by the following command:

```
>DESCRIPTOR SEQUENCE 12, 15, 16, 32*  
DESCRIPTOR SEQUENCE 12, 15, 16, 32*  
>DICTIONARY STRUCTURE*  
DICTIONARY STRUCTURE*
```

The DESCRIPTOR SEQUENCE command used in conjunction with DICTIONARY STRUCTURE command restricts output of the DICTIONARY STRUCTURE command to just those descriptors whose sequence numbers appear in the former. The DICTIONARY STRUCTURE COMMAND prints the name and complete specification of the requested descriptors. The output of these commands is:

12. GENERAL NUMERICAL METHOD
 OPTION CODE NUMBER OF CHARACTERS IN LONGEST STATE 32

CODE NAME

- 1 FINITE DIFFERENCE
- 2 FINITE ELEMENT
- 3 MONTE CARLO
- 4 FINITE DIFFERENCE/FINITE ELEMENT

15. MAXIMUM SPATIAL DIMENSIONALITY
 OPTION CODE NUMBER OF CHARACTERS IN LONGEST STATE 17

CODE NAME

- 1 ONE DIMENSIONAL
- 2 TWO DIMENSIONAL
- 3 THREE DIMENSIONAL

16. SPATIAL GEOMETRY
 OPTION CODE NUMBER OF CHARACTERS IN LONGEST STATE 33

CODE NAME

- 1 RECTANGULAR
- 2 CYLINDRICAL
- 3 SPHERICAL
- 4 RECTANGULAR/CYLINDRICAL
- 5 RECTANGULAR/SPHERICAL
- 6 CYLINDRICAL/SPHERICAL
- 7 RECTANGULAR/CYLINDRICAL/SPHERICAL
- 8 SPECIAL TREATMENT

32. TYPE OF FLUID FLOW
 OPTION CODE NUMBER OF CHARACTERS IN LONGEST STATE 23

CODE NAME

- 1 INVISCID COMPRESSIBLE
- 2 VISCID COMPRESSIBLE
- 3 INVISCID INCOMPRESSIBLE
- 4 VISCID INCOMPRESSIBLE
- 5 NONE

The user can now make an intelligent query as to number of codes in the data base which use a finite difference method and which calculate two dimensional cylindrical inviscid compressible flows.

```
>HOW MANY HAVE GENERAL NUMERICAL METHOD,FINITE DIFFERENCE AND
  HOW MANY HAVE GENERAL NUMERICAL METHOD,FINITE DIFFERENCE AND
>MAXIMUM SPATIAL DIMENSIONALITY,TWO DIMENSIONAL AND
  MAXIMUM SPATIAL DIMENSIONALITY,TWO DIMENSIONAL AND
>TYPE OF FLUID FLOW,INVISCID COMPRESSIBLE AND
  TYPE OF FLUID FLOW,INVISCID COMPRESSIBLE AND
>SPATIAL GEOMETRY,CYLINDRICAL*
  SPATIAL GEOMETRY,CYLINDRICAL*
```

The response for this query is:

ISOLATIONS	TOTAL	PERCENTAGE
2	20	10.00

Notice that the number of hits, the total number of data base items and the percentage of hits to total items is always displayed.

Now wishing to see the code names for the two codes satisfying the above query we ask:

```
>PRINT CODE NAME FOR WITH HOLD*
  PRINT CODE NAME FOR WITH HOLD*
```

The HOLD instruction is used so that the long Boolean expression need not be repeated. The result of this query is:

```
BLAST
LASXPT
```

ISOLATIONS	TOTAL	PERCENTAGE
2	20	10.00

Finally, wishing to see more information about the codes BLAST and LASXPT, we issue the following sequence of commands:

```
>INDENT 0*
  INDENT 0*
```

```
>PRINT 1,2,3,4,5,7,8,20,24,25,34,41,42,43,44 FOR WITH 1,BLAST OR
  PRINT 1,2,3,4,5,7,8,20,24,25,34,41,42,43,44 FOR WITH 1,BLAST OR
>1,LASXPT*
  1,LASXPT*
```


The INDENT 0 command instructs the system to indent zero spaces (no indentation) between outputs. The PRINT command illustrates that the numeric value of a descriptor may be used in place of its name. The results of these instructions are:

BLAST
 UNSTEADY/2D/EULERIAN/SINGLE MATERIAL/FINITE DIFFERENCE/COMPRESSIBLE FLUID
 CALCULATION OF MUZZLE BLAST FLOW FIELDS/TP-4155 PICATINNY ARSENAL/AD881
 523/DEC 1970
 T D TAYLOR
 FORTRAN
 OPERATIONAL/EASY TO RUN
 MUZZLE BLAST CALCULATIONS
 PERFECT GAS LAW
 REFLECTIVE/FREE-SLIP
 70 by 70
 NONE
 C K ZOLTANI
 1
 EVALUATION OF THE COMPUTER CODE BLAST DORF HELP AND HEMP FOR SUITABILITY
 OF UNDEREXPANDED JET FLOW CALCULATION BRL1659
 C K ZOLTANI
 LASXPT
 NONEQUILIBRIUM/RADIATION-HYDRODYNAMICS/ATMOSPHERIC TRANSPORT AND RESPONSE/
 PLASMA CHEMISTRY/LASER PLASMA/LASER TARGET
 THE BRL NONEQUILIBRIUM LASER PLASMA-TARGET INTERACTION CODE/BRL DRAFT REPORT
 JOSEPH LACETERA
 FORTRAN
 OPERATIONAL/COMPLICATED TO RUN
 LASER PLASMA INTERACTIONS
 PERFECT GAS LAW
 TRANSMITTIVE/MOVING
 50 BY 5
 NONEQUILIBRIUM
 JOSEPH LACETERA/CONTINUUM MECHANICS CENTER/BRL/APG MD/301 278 4353
 2
 LASXPT 1 PLASMA INTERACTIONS/BRL DRAFT REPORT
 JOSEPH LACETERA

ISOLATIONS	TOTAL	PERCENTAGE
2	20	10.00

These examples have been constructed as an illustrative group and are not meant to be complete or portray all of the capabilities of the system. For example, the reverse numerical and alphabetical sorting capabilities have not been shown.

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

3 OF 7
ADA
046552



VI. DISCUSSION

This report deals with the CMC's computer code data base which uses CREATABASE, a relational data base analysis system. Besides describing the data base, the manner in which it is accessed and queried is also explained and corresponding examples are given.

CMC personnel are not only the designers of the data base but also are its primary users. Care and maintenance of the data base is one of the CMC's functions. In addition to ensuring correctness of the current data, the center will add new data as it becomes available. Such data is not limited to that defined by the current 62 descriptors since new descriptors will be added as required. No attempt will be made to make the data base a complete reference system for all codes. However, the CMC will consider and catalog not only the most promising, but also the most used codes.

In addition to the data base itself, the CREATABASE system must undergo change and not remain a static inflexible tool. One area in which CREATABASE can be improved is that of subtabling. The amalgamation of like descriptors (for example, authors) into a single descriptor will allow for easier querying and new relations to be formed. For instance, the output of a query involving an author may produce his co-authors for a single reference or all his co-authors for all his published works. Furthermore, short queries involving a single amalgamated descriptor are preferable to long descriptor lists or Boolean expressions. Finally, the amalgamated descriptor will alleviate some of the need for handling subset binary files through the fortuitous production of information. Another area in which CREATABASE can be improved involves limited alphanumeric searching for name descriptors. Such a change would not only extend the textual capabilities of CREATABASE but free the user from entering artificial data for several types of applications and/or exactly specifying a descriptor state in the Boolean expression. The form for this extension should also provide for a range of values rather than a specific state. Other improvements of CREATABASE are possible; however, the two items listed above will make this system even better.

Finally, the use of this data base is encouraged as an information retrieval system. Furthermore, comments and suggestions on its structure and contents are welcome.

ACKNOWLEDGEMENT

The authors wish to thank Daniel Analytical Services Corporation for permission to publish Figure 2.

APPENDIX I. QUESTIONNAIRE USED TO GATHER DATA
BASE DATA

QUESTIONNAIRE

1. The name of the code (acronym plus its meaning) is _____.
2. List the following information on the user's manual:
Author(s) _____
Address of Authors _____
Title _____
Report Number _____
Date of Publication _____.
3. Code is operational on the following computers:
CDC 7600 UNIVAC 1108 BRLESC _____.
4. The following people are knowledgeable in the code's use:

5. Primary application of the code is _____.
Second application of the code is _____.
Tertiary application of the code is _____.
6. The type of mesh used is
Eulerian Lagrangian Eulerian & Lagrangian unknown.
7. The code uses the following general numerical method:
finite difference finite element Monte Carlo _____.
8. The code uses the following particular numerical method:
characteristics Lax-Wendroff random walk Galerkin multipass integral

_____.
9. The order of the numerical scheme is
_____ not applicable unknown.
10. The code performs unsteady calculation: yes no.
11. The code can treat the following spatial geometry(ies):
rectangular cylindrical spherical.

12. The code can treat the following spatial dimensionality:

one two three.

13. List the variables computed directly from the transport equations.

14. The code can use the following equations of state:

Tillotson perfect gas law BRLGRAY JWL CHARTD PUFF

_____.

15. The code can apply the following types of boundary conditions:

reflective transmittive non-slip free-slip moving free surface

_____.

16. The maximum grid size for the code is _____ by _____.

17. The code has the following type of rezoning capability:

automatic manual none unknown _____.

18. The code does the following type of radiation transport

equilibrium non-equilibrium none unknown.

19. The code does the following type of energy deposition:

time-independent time-dependent none unknown _____.

20. The code does the following type of chemical reactions:

equilibrium non-equilibrium none unknown.

21. The code does the following type of atomic reactions:

equilibrium non-equilibrium none unknown _____.

22. The code calculates material response: yes no.

23. The code treats solids as an elastic plastic:

yes no not applicable unknown .

24. The code has an interface capability:
- yes no not applicable unknown.
25. The code can handle _____ (number of) different materials.
26. The code treats the following types of fluid flow:
- inviscid compressible viscous compressible inviscid incompressible
- viscous incompressible none unknown.
27. The code treats shocks by the following method:
- artificial viscosity shock fitting _____
- none not applicable.
28. The code solves the following equations:
- conservation of mass conservation of momentum conservation of energy
- Boltzmann's equation _____ _____.
29. The code is written in the following computer language(s):
- FORTRAN ALGOL APL
30. The code has the following special features:
- strength option tracer particles combustion option sliplines
- _____ _____ _____.
31. The following reports contain information relating to the code itself or the code's performance. For such reports give author(s), report number(s), and title(s) or key words. The total description per report should be less than 120 characters including blanks.
- 1.
 - 2.
 - 3.
 - 4.
 - 5.
 - 6.
 - 7.
 - 8.
 - 9.

32. Describe the salient features of the code in less than 120 characters; for example,

HELP: unsteady, 2D, Eulerian, multi-material, finite difference, integral formulation, solid and compressible fluid applications.

33. Please list any pertinent computer code properties omitted and any other comments.

APPENDIX II. THE DATA BASE

193

```

NON-
3D HEAT TRANSFER/HEAT CONDUCTION/HEAT DIFFUSION/MONTE CARLO/GREENS FUNCTIONS/HOMOGENEOUS BC/INHOMOGENEOUS BC
SOLUTION OF THE TIME-DEPENDENT HEAT CONDUCTION EQUATION IN COMPLEX GEOMETRY BY THE MONTE CARLO METHOD
E E THOUBETZKOY/M KALOS/H STEINBERG//JNLCH-325/DEC 1976
FORTAN
CDC 7600 HUNTSVILLE
OPERATIONAL/EASY TO RUN
HEAT TRANSFER
---
EULERIAN
MONTE CARLO
RANDOM WALK
---
THREE DIMENSIONAL
RECTANGULAR/CYLINDRICAL/SPHERICAL
YES
HEAT CONDUCTION
TEMPERATURE
---
---
REFLECTIVE/TRANSMITTIVE/FREE SURFACE
---
SPECIAL
NOT APPLICABLE
NO
NO
100
YES
NONE
NOT APPLICABLE
NONE/EQUILIBRIUM
TIME-DEPENDENT
NONE
NONE
---
TRANSLATIONALLY REPETITIVE ARRAYS
---
J KLEM/N E BANKS
0
---
---
---
---
---
---
---
---
---
---
---
---

```


198

FINISHING UNIT
 NONEQUILIBRIUM/RADIATION-HYDRODYNAMICS/ATMOSPHERIC TRANSPORT AND RESPONSE/ATOMIC IONIZATION/FIREBALL CHEMISTRY/
 THE BRL NONEQUILIBRIUM NUCLEAR FIREBALL CODE/BRL DRAFT REPORT/
 JOSEPH LACETERA ET.AL./CONTINUUM MECHANICS CENTER/BRL/APG MD/301 278 4353
 FORTRAN

-DC 7600 HUNTSVILLE
 OPERATIONAL/COMPLICATED TO RUN
 FIREBALL PHENOMENOLOGY
 4-RAY TRANSPORT
 SPHERICAL BLAST
 EU-ERIAN
 -EVITE DIFFERENCE
 VARIATION OF PIC
 FIRST
 ONE DIMENSIONAL
 SPHERICAL
 YES
 CONSERVATION OF MASS/MOMENTUM/ENERGY/PARTICLES
 DENSITIES/VELOCITIES/GAS ENERGY/RADIATION ENERGY
 PERFECT GAS LAW

TRANSMITTIVE/MOVING
 50 BT 1
 MANUAL/AUTOMATIC
 YES

YES
 NOT APPLICABLE

1

NOT APPLICABLE
 INVISCID COMPRESSIBLE

SPECIAL TREATMENT

NONEQUILIBRIUM

TIME-DEPENDENT PHOTON ENERGY DEPOSITION

NONEQUILIBRIUM

NONEQUILIBRIUM

NONEQUILIBRIUM

NONEQUILIBRIUM IONIZATION/NON-LTE

NOT ROUTINELY RUN

JOSEPH LACETERA

1

EFFECTS OF NONEQUILIBRIUM PHENOMENA ON RADAR TRANSMISSION/BRL DRAFT REPORT

JOSEPH LACETERA/G DAUM

EPIC-3

TO BE PUBLISHED AS URL CONTRACT REPORT ON OR ABOUT MAY 1977
S R JOHNSON/DEFENSE SYSTEMS DIV/HONEYWELL INC/MINNEAPOLIS MN
FORTHAN

JHIVAC 1100 EDGEWOOD

PILOT

KINETIC ENERGY PENETRATION

SHAPED CHARGE STUDIES

IMPACT AND WAVE PROPAGATION

AGRAMGIAN

FIVITE ELEMENT

NOT APPLICABLE

THREE DIMENSIONAL

RECTANGULAR/CYLINDRICAL/SPHERICAL

YES

WIE-GRUNEISEN

REFLECTIVE/TRANSMITTIVE/NON-SLIP/FREE-SLIP/MOVING/FREE SURFACE
USES SUBSTRUCTURE METHOD---GRID LIMITED ONLY BY TIME-COST CONSIDERATIONS

NONE

NO

YES

YES

5

YES

VISCID COMPRESSIBLE

ARTIFICIAL VISCOSITY

NONE

NOT APPLICABLE

NONE

NONE

NONE

STRENGTH OPTION/SPLINES

J A ZUKAS/M S CHAWLA/G H JONUS/B RINGERS

0

141 L
 UNSTEADY/2D/4 EULERIAN/MULTI-MATERIAL/FINITE DIFFERENCE/INTEGRAL FORMULATION/PIC TYPE/SOLID AND FLUID APPLICATIONS
 HELP A MULTI-MATERIAL EULERIAN.../VOLUMES 1-2/SSSR-73-2654 MAY71/HRLCR39 JULY75
 1 J HAGGPAU D F WILKINS ET.AL./SYSTEM SCIENCE AND SOFTWARE LAJOLLA CA 92036
 2 FORTRAN
 3 RLSC/CIC 7600 HUNTSVILLE/UNIVAC 1108 EDGEWOOD
 4 OPERATIONAL/EASY TO RUN
 5 SHAPED CHARGE STUDIES
 6 KINETIC ENERGY PENETRATION
 7 FRAGMENTATION MUNITIONS
 8 EULERIAN
 9 FINITE DIFFERENCE
 10 VARIATION OF PIC
 11 FIRST
 12 TWO-DIMENSIONAL
 13 RECTANGULAR/CYLINDRICAL
 14 YES
 15 CONSERVATION OF MASS/MOMENTUM/ENERGY
 16 DENSITY/VELOCITIES/TOTAL ENERGY
 17 TILLOTSON
 18 JNL-JONES/WILKINS/LEE
 19 PERFECT GAS LAW
 20 ---
 21 REFLECTIVE/TRANSMITTIVE
 22 ON COD 7600 60 BY 200//ON UNIVAC 1108 34 BY 86
 23 MANUAL
 24 NO
 25 YES
 26 YES
 27 23
 28 YES
 29 INVISCID COMPRESSIBLE
 30 ARTIFICIAL VISCOSITY
 31 NONE
 32 TIME-DEPENDENT EXPLOSIVE DETONATION
 33 NONE
 34 NONE
 35 -LIMITED
 36 STRENGTH OPTION/TRACER PARTICLES/SPLINES
 37 OVER THERMALIZATION
 38 5 JONES/J MISEY/M CHAWLA/JANET LACETERA/JOE LACETERA/J HARRISON/J SCHMITT
 39 4
 40 RLSC AN ADVANCED EULERIAN CODE FOR PREDICTING SHAPED CHARGES VOLUME1
 41 SYSTEMS SCIENCE AND SOFTWARE LAJOLLA CA 92036
 42 NUMERICAL ANALYTICAL AND EXPERIMENTAL INVESTIGATION OF PENETRATION BY KINETIC PROJECTILES AFATL-TR-72-48
 43 SYSTEMS SCIENCE AND SOFTWARE LAJOLLA CA 92036
 44 EFFECTS OF HIGH EXPLOSIVE PARAMETERS AND DEGREE OF CONFINEMENT ON JETS FROM LINED SHAPED CHARGES BRL CR 245
 45 SYSTEMS SCIENCE AND SOFTWARE LAJOLLA CA 92036
 46 RESEARCH STUDY AND ANALYSIS FOR IMPROVEMENT OF SHAPED CHARGE CODE BRLCR140
 47 SYSTEMS SCIENCE AND SOFTWARE LAJOLLA CA 92036
 48 ---
 49 ---
 50 ---
 51 ---
 52 ---
 53 ---
 54 ---
 55 ---
 56 ---
 57 ---
 58 ---
 59 ---
 60 ---

UNSTEADY/1D/TWO PHASE/FINITE DIFFERENCE/INVISCID COMPRESSIBLE GAS/FLAMESPREADING MOBILE BED RHEOLOGY
 NUMERICAL ANALYSIS OF A TWO PHASE FLOW WITH EXPLICIT INTERNAL BOUNDARIES PGA-TR-76-2 SEPT 76
 S GOUGH PO BOX 1614 PORTSMOUTH NH

FORTHAM
 JNIVAC 1108 EDGEWOOD
 OPERATIONAL/FAST TO RUN
 FLAMESPREADING IN GUNS

EULERIAN
 FINITE DIFFERENCE
 MACCORMACK
 SECOND
 ONE DIMENSIONAL

YES
 CONSERVATION OF MASS/MOMENTUM/ENERGY
 POROSITY/GAS MASS/PHASE VELOCITIES/GAS ENERGY
 PERFECT GAS LAW
 NOBLE-ADCL

REFLECTIVE/TRANSMITTIVE/MOVING

50

NONE

NO

NOT APPLICABLE

2

NO
 INVISCID COMPRESSIBLE
 SHOCK FITTING
 NOVE
 PROPELLANT COMBUSTION

NONE

NONE

EXTENSIVE
 COMBUSTION OPTION

W NELSON/R D ANDERSON

1

COMPARISON OF THREE TWO PHASE FLOW CODES BRL MR2729 JAN77

W NELSON

4ETHIC
 JUSTIAUT/30/EULRIAN/MULTI-MATERIAL/FINITE DIFFERENCE/INTLGRAL FORMULATION/SOLID AND COMPRESSIBLF FLUID APPLICATIONS
 DEVELOPMENT OF URUNANCE VELOCITY MULTIMATERIAL 3D PERFORATING CODE FOR FINITE PLATES SSS R 76-2861(DRAFT) FEB 76
 J HAGEMAN/E P LEL SYSTEMS SCIENCE AND SOFTWARE LAJOLLA CA 92036
 FORTNAN

JUIVAC 1102 EDGEWOOD

--- GENETIC ENERGY PENETRATION

--- EULERIAN

--- FIVITE DIFFERENCE

--- FIRST

--- THREE DIMENSIONAL

--- RECTANGULAR

--- YES

--- CONSERVATION OF MASS/MOMENTUM/ENERGY

--- FILLTOSOM

--- REFLECTIVE/TRANSMITTIVE

IN HY 27 HY 13

--- NONE

--- YES

--- YES

--- YES

--- INVISCID INCOMPRESSIBLE

--- ARTIFICIAL VISCOSITY

--- NONE

--- NOT APPLICABLE

--- NONE

--- NONE

--- STRENGTH OPTION/TRACER PARTICLES/CAN HANDLE AS MANY DIFFERENT MATERIALS AS DESIRED

CODE IS UNTRIED/GRID MUST HAVE AT LEAST 3 ROWS IN EACH DIRECTION/BOUNDARIES MUSTINITIALLY COINCIDE WITH CELL BOUNDARIES

3 H JOHAS/J MISEY

0

```

SHILL
-----
SITE-A SECOND ORDER EULERIAN CODE FOR HYDRODYNAMIC AND ELASTIC-PLASTIC PROBLEMS BRL CR REPORT 255 AUG 75
S S Z HURSTEIN/H SCHECHTEN/ZE TURNEL/MAGI INC ELMSFORD NY
FORTHAN
JNIVAC 1108 EDGEWOOD
PILOT
KINETIC ENERGY PENETRATION
SHAPED CHARGE STUDIES
---
EULERIAN
EULYTIC DIFFERENCE
-AX-WENDROFF
SECOND
TWO DIMENSIONAL
RECTANGULAR/CYLINDRICAL
YES
CONSERVATION OF MASS/MOMENTUM/ENERGY
---
FILLITSON
MIL-GHUNEISEN
---
REFLECTIVE/MOVING/FREE SURFACE
109 BY 30
NONE
NO
YES
YES
5
YES
INVISCID COMPRESSIBLE
ARTIFICIAL VISCOSITY
NONE
NOT APPLICABLE
NONE
NONE
NONE
STRENGTH OPTION
---
J A ZUKAS
3
3RL CONTRACT REPORT NO 294
---
3RL CONTRACT REPORT NO 295
---
3RL CONTRACT REPORT NO 175
---
---
---
---
---
---
---
---
---
---
---

```



```

RIPPLE
UNSTEADY/2D/EULERIAN/ONE MATERIAL/FINITE DIFFERENCE/INVISCID COMPRESSIBLE FLOW
RIPPLE-A 2D UNSTEADY EULERIAN HYDRODYNAMIC CODE-USERS MANUAL DRL REPORT 1632 FEB 73
C # NELSON
FORTRAN
BULESC/UNIVAC 1108 EDGEWOOD
OPERATIONAL/EASY TO RUN
BLAST WAVES
---
EULERIAN
FINITE DIFFERENCE
FLUID IN CELL
---
140" DIMENSIONAL
RECTANGULAR/CYLINDRICAL
YES
CONSERVATION OF MASS/MOMENTUM/ENERGY
DENSITY/VELOCITIES/INTERNAL ENERGY
TILLOTSON
PERFECT GAS LAW
---
REFLECTIVE/TRANSMITTIVE
50 HY 50
NONE
YES
NO
1
INVISCID COMPRESSIBLE
ARTIFICIAL VISCOSITY
NONE
TIME-DEPENDENT
NONE
LIMITED
---
C # NELSON
1
NUMERICAL CALCULATIONS OF FLOW FROM RECOILLESS RIFLE NOZZLES BRL REPORT 1688
C # NELSON
---

```

A PARALLEL ARRAY COMPUTER FOR THE SOLUTION OF FIELD PROBLEMS

W.R. Cyre*, C.J. Davis*, A.A. Frank*,
L. Jedynak*, M.J. Redmond*, and V.C. Rideout*

I. INTRODUCTION

The von Neumann single-processor digital computer [1] has dominated the computer scene for many years, with improvements in speed and memory capability continuing to appear, accompanied by reductions in size and cost. However, the speed possible in such machines is bounded by limits on the speed of signal transmission, and even the fastest and most powerful single processor or serial machines have been found to have severe limitations when applied to the simulation of large dynamic problems such as the earth's weather and climate [2,3,4]. The adoption of a limited amount of parallelism in machine architecture has not overcome these difficulties, and in some cases has led to an increase in software problems. Furthermore, attempts made to replace the now expensive analog-hybrid machines by serial digital computers for simulation of large dynamic systems have revealed that serious inadequacies in speed and in user-machine interaction remain.

A new parallel array computer is proposed in this paper, of a design based on the needs of those concerned with large scale simulation. This design uses the small

* The Department of Electrical and Computer Engineering,
University of Wisconsin-Madison, Madison, WI 53706

and low-cost but powerful microprocessors and memory chips which are now available. The array computer can be configured to have some degree of isomorphism with the physical problem being solved, as in the parallel analog computer. It is felt that such a computer using a large number (N) of such microcomputers can be designed to keep software simple, while approaching a speed advantage of N times over the single processor machine [22].

II. APPLICATIONS OF PARALLEL ARRAY COMPUTATION

Array computers have been classified by Flynn [5] as being either SIMD (single-instruction stream, multiple-data stream) or MIMD (multiple-instruction stream, multiple-data stream) machines. The best-known machine of the SIMD type is the ILLIAC IV machine [6,7,8]. This machine was based on a previously proposed machine, the SOLOMON computer [9], which was conceived with the weather simulation problem in mind [10].

The ILLIAC IV and most other parallel array machines appearing in the literature have a limited number of processors such that to expand apparent array size, some serial-parallel approach to large problems is necessary, which leads to software complexity [11,12,13]. The advent of the microprocessor and IC memory technologies makes it possible to propose a more versatile MIMD type of parallel array machine with enough microcomputers to assign one to each node of a discretized partial differential equation simulation (or one to each state variable of a system simulation) in most problems in the classes of problems for which the machine was built. Where absolutely necessary more than one equation could be assigned to any node with some degradation in performance. Such a machine,

the Wisconsin Parallel Array Computer (WISPAC), is proposed, which is capable of being extended to a three-dimensional array of as many as 100x100x20 or 200,000 processors, each connected to its six nearest neighbors, together with a scheme (see Sec. III) for fast "pass-through" interconnection paths among non-adjacent as well as adjacent processors. This parallel array computer could be used in the study of a large class of simulation problems, as listed below:

- (1) Finite difference solution of various field problems described by partial differential equations (PDEs)

Here the more challenging dynamic problems would be of principal concern, and the basic three-dimensional array could be software-configured to also accommodate problems in one or two dimensions. In addition to problems of weather and climate simulation, referred to above, geophysical, plasma confinement and fluid dynamics problems could be handled. The considerable increase in speed possible because of parallelism should make available some degree of on-line interactive operation in large PDE problems, as well as parameter and initial condition estimation.

- (2) Solution of PDE problems by finite element techniques

Finite element methods [14] have certain advantages over finite difference schemes, and have been valuable in certain static problems in elasticity, particularly in cases where the boundaries are irregular. The form ultimately taken by a finite element formulation is much like that for a finite difference formulation, so that an array machine may be equally useful if this

method of solution is used.

(3) Monte Carlo solution of PDEs

It has been shown that the conditional probability function of a Markov process [15] satisfies a second-order elliptic or parabolic PDE and this has led to suggestions that this method be used in problems with difficult boundary shapes. In Monte Carlo solutions the potential at a point is determined by averaging the potential encountered at a boundary after a large number of random walks started at the point and terminating at the first boundary encountered. Serial digital methods for solution by this approach have not been satisfactory, because of the time needed to make 1000 or more runs for each point in the discretized field, and attempts to use the hybrid computer [16] also have not led to success. With a parallel array computer, runs could be made simultaneously from all M points of interest in the field, and a speed advantage over the serial computer of M times might be approached.

(4) Continuous System Simulation

Continuous dynamic systems, ordinarily described by sets of first-order ordinary differential equations in the state variables, have often been studied using analog-hybrid computers, particularly in the aerospace industry. As in the case of PDE boundary value problems, the parallel nature of these problems, as well as requirements on speed of solution make the parallel array computer a potentially useful tool for simulation studies.

The serial digital computer is now being used for simu-

lation of dynamic systems, sometimes with programming aids such as CSMP (Continuous System Modeling Program) [17,18]. Although increased speeds have been provided by serial machines, the sizes of problems of interest have grown even faster. Particularly in the study of socio-economic systems, electric power systems, ecologic-environmental systems and physiological systems, the size of the system makes the on-line operation of dynamic simulations both slow and expensive. Without the possibility of user interaction, simulation tends to be so slow that it is often much less useful.

A parallel array computer of the kind proposed and consisting of 100 to 200 microcomputers would be adequate for most system simulations now being studied by CSSL or hybrid computer simulation techniques (as well as small PDE-described systems). Solution of the equation for each state variable would be assigned to one microcomputer in the array, with the interconnection set up by software, as described in IV, below.

As in the case of PDEs, speed requirements are greatly increased if the needs of optimization or parameter estimation make repeated simulation runs a necessity [19]. The speed advantage of a parallel array machine would also be useful in system problems involving stochastic signals or parameters.

(5) Discrete System Simulation

Although discrete systems such as those arising in simulation study of transportation problems, queing and cell-growth are better handled by serial machines than on continuous systems, the increasing size of

these problems may also make for difficulties in the future. Combined continuous-discrete systems [20] offer even more of a challenge, especially if combined with interactive simulation requirements. The proposed computer is as well-suited to simulation of discrete and combined systems as it is to the simulation of continuous systems.

(6) Data Manipulation

The manipulation and reduction of data in such applications as pattern recognition, radar signal processing, Fast Fourier Transforms, etc., may also be more effectively carried out by parallel array processors, and indeed machines have been devised for such purposes [21,22]. The proposed array computer can also be adapted to these applications, which are now being attempted with simpler array machines.

III HARDWARE FEATURES OF WISPAC

The proposed Wisconsin Parallel Array Computer will employ a basic two-level architecture which can be expanded to three levels for larger systems. The individual node processors which form the computing array are at the base level of the hierarchy. As previously stated, node processors are arranged in a three dimensional cubical configuration where each node processor (NP) is connected to its six nearest neighbors. Each NP contains a full microprocessor CPU with instruction-decode logic, arithmetic-logic unit (ALU), addressing logic and I/O capability. Each NP also supports the direct communication linkages to its six neighbors in the three-dimensional array. Opposite array edges are linked or wrapped around to form a hardware wired "hyper-torus". Actual wrap-around

configuration, and in fact all node-interconnections, are under software control. A block representation of an NP is shown in Fig. 1. Each node processor supports its own local data and program storage memory. This differs from the SOLOMON [9] and ILLIAC [6] type parallel processors where each processing element can directly use local memory only for data storage. Thus, the Wisconsin Parallel Array Computer has a Multiple Instruction Stream -- Multiple Data Stream, or MIMD [7], architecture, such that any number of node processors may execute different instructions simultaneously.

The local memory associated with each node processor is also integrated into the communication system between the node processors and the next level of the hierarchy, the sector control computer. A sector may be defined as a set of from $5 \times 5 \times 5$ to $25 \times 25 \times 20$ node processors. A diagram of the relationship of one node processor to its local memory and to the sector control computer (SCC) is shown in Fig. 2. The sector control computer supports its own data and program memory. However, the local memory of any node processor can be appended to the end of the SCC's dedicated memory to allow memory-to-memory transfers between the two memory blocks. In this manner local programs may be loaded into the array and I/O may be achieved during array operation. The memory communication bus can also be routed for external access to the sector. This allows multiple sector expansion of the array.

Memory cost will be the largest portion of overall array cost therefore memory word width and size for each NP must be optimized based on a study of the systems to be simulated in the computer. Error correcting memory must be considered for the node processors to insure

reasonable reliability. Error detection and correction for the node processor CPUs must also be studied. One possible method for CPU error detection is for each NP to support redundant CPUs. For correction, the NPs and the sector control computer should be software compatible. When a faulty NP is detected, the SCC can execute the NP's tasks directly through the appended memory communication system.

The array computer can be expanded by adding array sectors. Array continuity is maintained across sector boundaries setting up nearest neighbor linkages between node processors at sector edges. In the present design of WISPAC each fully expanded sector can support up to $25 \times 25 \times 20$ node processors. A full complement of sixteen sectors would bring the maximum array size to $100 \times 100 \times 20$. When more than one sector is used, it is necessary to add a master array computer as a third level to the hierarchy for control of overall array utilization. Under master computer control, several problems could simultaneously be run, each in a different sector, or several in one sector. (This function could also be realized within a single sector under sector computer control.) The array could also be fully utilized by a single problem. An overall block diagram is shown in Fig. 3. (Note that the node memory bus diagram is a simplified representation of that shown in Fig. 2.)

If intra-array communication is hardware limited to next nearest neighbor linkages, the classes of problems that can be easily solved is somewhat restricted. Therefore a second kind of intra-array communication is proposed to allow a limited number of high-speed arbitrary path linkages between any pairs of NPs in the array. The scheme

integrates a commutation capability into the nearest neighbor linkages. An output linkage on one of the six node processor sides could be "switched" either to carry data from the output port of the microprocessor associated with that side, or data from any one of five input linkages (the input linkage on the same side as the output linkage is excluded). Diagrams of some possible configurations for commutation are shown in Fig. 4 (here two-dimensional representations are used, for simplicity). A simulation problem utilizing commutation (in two dimensions) is diagrammed in Fig. 5 of Section V. Commutation operation could be controlled by the microprocessor at each node so that commutation could be changed during program runs. The sector control computer could also control commutation globally. With this scheme, a limited number of arbitrary paths could be constructed between any two node processors.

IV SOFTWARE CONSIDERATIONS AND IMPLEMENTATION

Given a rectangular three-dimensional array of independently programmable microprocessors, each connected to its six nearest neighbors, the programming and solution of a field problem such as the diffusion equation by finite differences [23,24] is straightforward. If the basic equation is

$$\frac{\partial \phi}{\partial t} = K \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} \right) \quad (1)$$

then the "molecule" of Fig. 5 gives the difference form:

$$\frac{\phi_{0,j+1} - \phi_{0,j}}{\Delta t} = \frac{K}{h^2} (\phi_{E,j} + \phi_{W,j} + \phi_{N,j} + \phi_{S,j} + \phi_{U,j} + \phi_{D,j} - 6\phi_{0,j}) \quad (2)$$

The explicit formulation will give a new value of ϕ at each point in one calculation. However, an implicit formulation, such as the one shown below, may be preferred in many cases because it permits larger steps in time.

$$\frac{\phi_{0,j+1} - \phi_{0,j}}{\Delta t} = \frac{K}{2h^2} (\phi_{E,j} \dots - 6\phi_{0,j}) + \frac{K}{2h^2} (\phi_{E,j+1} \dots - 6\phi_{0,j+1}) \quad (3)$$

This is the Crank-Nicolson formulation, and requires some implicit form of solution. Because the matrix is sparse, iterative methods of solution such as the Jacobi scheme are used. Such methods may be applied in the parallel array computer by requiring each microcomputer to simultaneously calculate an estimate of $\phi_{0,j+1}$ (using $\phi_{E,j+1}$ estimates and $\phi_{E,j}$, etc.), then exchange estimates $\phi_{0,j+1}$ etc., and determine new estimates until a convergence criterion is satisfied. In certain regular problems the number of iterations for convergence might be determined by a computation in the master computer: in more complex cases a convergence criterion might need to be incorporated in the programs of some or all of the microprocessors.

Hardware may be required to detect overall convergence in iterative solutions in order to maintain solution speed. One possible definition of global convergence could be the logical "AND" of all local convergences.

It should be noted that some of the schemes used to improve the speed of solution of field problems in serial digital computers (Gauss-Seidel, Alternating Direction Methods, etc.) may not be suitable for use in

a parallel array machine, which depends on parallelism to attain its speed advantages.

As previously stated, the proposed WISPAC is a Multiple Instruction Stream -- Multiple Data Stream parallel array computer in which the node processors can be individually programmed as von Neumann machines. Compilation and assembly of programs will take place at the sector control or master control computer levels.

Programming of any one node processor through the sector or master computers is a fairly simple task using FORTRAN, APL or some other user-oriented language. However, to program an entire array without requiring program development for each node requires that procedure-oriented programming tools be employed.

To implement the solution of a field problem in a homogeneous medium, a single program describing the medium at each point could be developed on the master or sector control computer and then filled into the NPs that represent the medium. If mathematical relationships can be formed for determining position, coefficients and/or formulation of solutions needed at boundaries it would be possible to develop boundary programs by using a procedure oriented FORTRAN or APL which could modify program statements, statement coefficients, and/or loading position into the array based on known relationships. If such relationships are difficult to state or do not exist, each NP or set of NPs at a boundary would have to be independently programmed. In similar fashion, programming the array computer for solution of the field in a non-homogeneous medium might be simplified by using known relationships.

If the geometrical configuration of the boundaries of the PDE equation are such that it fits well into a rectangular array, or into the cylindrical or toroidal array obtained by "wrap-around", then the geometrical interconnection problem is a simple one. If the shape does not fit well, one of these array forms may still be satisfactory when used with the aid of the pass-through or commutation capability. For a computer dedicated to some single large problem (such as weather or climate), a special array might be set up.

Other general-purpose array structures might be preferred for some parallel array computers of the kind described here. Thus if triangles are to be used in place of squares, each node processor would need to have six nearest neighbor communication channels in the two-dimensional case, and twelve in the three-dimensional case. Although the number of communication linkages required would be greater, the solution of some problems, especially those related to finite elements, would be simplified.

It should be noted that while special hardware consisting of extra links to neighboring processors may speed and simplify problem solution for cases where triangles must be used, software can be used to effect the same information transfers for a cubical array by routing through processors.

To solve simulation or modeling problems, each node could be assigned to do the calculations associated with one state variable. In system modeling, a transfer function list might be specified along with an interconnection list. The master or sector control computers could process the lists, assigning transfer functions to

available nodes and constructing interconnect paths utilizing the commutation capability described in the preceding section. System simulation could be handled in a similar fashion where state variables would be assigned to available nodes and interconnection paths constructed by software. The three-dimensional structure of the WISPAC greatly simplifies interconnection path definition for two-dimensional simulation and modeling problems. Using a processor node depth of 20 levels can be compared to constructing 20-layer printed circuit cards. Simple path construction is assured in most practical cases.

For the general simulation case, one method of overall problem control requires that each processor indicate the end of its computation cycle by setting a flag. The master would initiate I/O and control instructions after all node processors have signified that they are ready for the next iteration. Another more versatile method is suggested by Miller [22], in which computation proceeds upon availability of data to the particular node, i.e. a node "fires" when sufficient data is present. Advantages of this method are as follows:

1. There is less time spent idle when I/O lists of different length are present
2. The speed will not be less than the hardware-lockstep method, and may often be greater.
3. If software passing of information becomes necessary, no delay is encountered due to the lock-step process.
4. With the possibility of data-available computation, it becomes possible to have more than one user efficiently occupying space in the array at a given time (multi-programming as well as multi-processing).

Given the ability of each processor to process data out of step, the setting up of accumulation paths as well as software passing of information becomes easily possible. Such paths are also necessary in matrix multiplication and other applications, to overcome the limitation presented by the number of inputs available at a particular node.

In order to perform the previously stated tasks various specialized software would have to be developed to ease user programming through the master or sector computers. The following system software would be required:

(1) Language Processor

A high level language is needed which is capable of handling files and programs in a manner similar to subscripted variables. This type of language is necessary due to the large number of programs which are manipulated and placed in the NPs. Such language processors have been implemented on serial machines in the past.

(2) Cross Compiler

The task of this software entity is to accept programs in some user-oriented language and translate them to machine language programs for the node processors.

(3) Loader

The most complicated part of the systems software for the proposed MIMD configuration will be the loader routines. The loader will be responsible for;

1. Allocation of solution areas in the array of processors.
2. Maintenance of available and unavailable processor lists.
3. Actual depositing of programs into array memories.

Pattern recognition techniques may be necessary to fit the maximum number of jobs into the solution array. If the master or sector computer maintains lists of assigned and unassigned processors, it may place many jobs into the array at any given time, just as many users are allocated memory in today's timesharing systems. Even in the instances of a single-user installation, limited use of these techniques may be necessary in order to allocate around areas of hardware failure to provide for graceful degradation. Part of these responsibilities may be allocated to the operating system.

V. PROGRAMMING OF AN EXAMPLE

The simulation study of the dynamics of the cardiovascular system [26] is often carried out on an analog or hybrid computer, in order to achieve real-time or faster operation [27]. If "multiple-modeling" is used [28], the pressure-flow dynamics may be set up on the analog part of a hybrid computer, and the slower mass transport dynamics on the digital part. In Fig. 6(a) the circuit representation of the discretized fluid-flow equations [26] is shown for a simple model of the systemic part of the circulation. Typical equations for pressures, p , and flows, f , (for part of the aorta) are:

$$p_5 - p_6 = R_6 f_6 + L_6 \frac{df_6}{dt}$$

and

$$C_6 \frac{dp_6}{dt} = f_6 - f_7 - f_{11}$$

The mass transport equations, if a perfect mixing chamber is assumed, corresponding to each compliance, is shown in Fig. 6(b). Typical equations for concentrations γ , and mass flows f^* are:

$$\begin{aligned} f_6^* &= \gamma_5 f_6 \\ f_{11}^* &= \gamma_6 f_{11} \\ f_7^* &= \gamma_6 f_7 \\ \gamma_6 &= \frac{\int_0^t (f_6^* - f_{11}^* - f_7^*) dt + q_6^*(0)}{p_6 C_6 + q_{6u}} \end{aligned}$$

where $q_6^*(0)$ is the initial mass in the compartment, and q_{6u} is the unstressed blood volume in the corresponding arterial segment.

The pressure-flow dynamics of Fig. 6(a) may be set up in a two-dimensional part of an array computer such as the WISPAC, with one NP assigned to each state variable, as shown in Fig. 5(c), with help from a strong topological correspondence. Note that the NP which yields the state variable p_6 also gives f_{11} , which is not a state variable since it is given by $f_{11} = (p_6 - p_{11})/R_{11}$.

Note too that pass-through is needed in this case to bring all four branch connections to node p_8 . If more than four branches join at a node the two-dimensional form can only be used if some more involved pass-through techniques are used.

The mass transport model of Fig. 6(b) may be programmed on the parallel array computer as shown in Fig. 5(d), where a topological resemblance is again seen. This model requires inputs from the pressure-flow model of

Fig. 6(c), and in a three-dimensional computer could lie in a plane directly above or below it so that interconnections will be simple. In modeling of the transport of more than one kind of material, models of the form of Fig. 6(d) may be stacked in parallel planes.

With proper software development the interconnection and data transfer routines could be pre-determined by a program in the master computer, as suggested in the preceding section.

VI. CONCLUSIONS

The proposed computer, both as regards hardware and software, does not require any particularly new or difficult development. Error correction, fault tolerance and fault location will offer problems which must receive much attention particularly if any large array machine of this type is built.

Initial studies will be made on a 3-microprocessor (plus master computer) machine now being built.

VII. ACKNOWLEDGEMENTS

The assistance of the Wisconsin Alumni Research Foundation and the Engineering Experiment Station at Wisconsin in support of this work is gratefully acknowledged.

REFERENCES

- (1) Burks, A.W., H.H. Goldstine and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", June 28, 1946; Summarized in Datamation, Pt. 1, pp. 36-41, Sept. 1962.
- (2) Haltiner, G.J., "Numerical Weather Prediction", Wiley, (1971).
- (3) Holloway, J.L. Jr. and S. Manabe, 1971, "Simulation of Climatology by a Global General Circulation Model", Mon. Wea. Rev., 99, pp. 335-370; 1971.
- (4) Hahn, D.G., "Simulation of Climate using a Sophisticated General Circulation Model of the Atmosphere", Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky, Ed.; Pub. AICA, 1975.
- (5) Flynn, M., "Some Computer Organizations and Their Effectiveness", IEEE Transactions on Computers, Vol. C-21, Sept. 1972; pp. 948-960.
- (6) Barnes, G.H. et al., "The ILLIAC IV Computer", IEEE Trans. Comp., Vol. C-17, No. 8, pp. 746-757, Aug. 1968.
- (7) Kuck, K.J., "ILLIAC IV Software and Application Programming", IEEE Trans. Comp., Vol. C-17, No. 8, pp. 758-770; Aug. 1968.
- (8) Davis, R.L., "The ILLIAC IV Processing Element", IEEE Trans. Comp., Vol. C-18, No. 9, pp. 800-816, Sept. 1969.
- (9) Slotnick, D.L., W.C. Borck and R.C. McReynolds, "The SOLOMON Computer", 1962 Fall Joint Computer Conf., AFIPS Proc., Vol. 22, 1962.
- (10) Carroll, A.B. and R.T. Wetherald, "Application of Parallel Processing to Numerical Weather Prediction", Jl. Assoc. Comp. Mach., Vol. 14, No. 3, pp. 591-614; July 1967.
- (11) Hobbs, L.C., et al., "Parallel Processor Systems, Technologies and Applications", Spartan Books Inc., 1970.
- (12) Enslow, P.H. Jr. (Ed.), "Multi-processors and Parallel Processing", Wiley, New York.

- (13) Kuck, D.J., "Parallel Processing of Ordinary Programs", in Advances in Computers, Vol. 15; M. Rubinoff and M.C. Yorits, Eds., Academic Press, 1976.
- (14) Zienkiewicz, O.C., "The Finite Element Method in Engineering Science", McGraw-Hill, London, 1971.
- (15) Bharucha-Reid, A.T., "Elements of the Theory of Markov Processes and Their Applications", McGraw-Hill, New York, 1960.
- (16) Handler, Howard, "High-speed Monte Carlo Technique for Hybrid-Computer Solution of Partial Differential Equations", Ph.D. Thesis, University of Arizona, 1967.
- (17) "System/360 Continuous System Modeling Program (CSMP) User's Manual: Program #360-A-CX-16X", IBM Corporation, White Plains, New York, 1969.
- (18) "The SCI Continuous Systems Simulation Language (CSSL)" J.C. Strauss, Ed.; Simulation, Dec. 1967.
- (19) Rideout, V.C. and J.E. W. Beneken, "Parameter Estimation applied to Physiological Systems", Proc. AICA, Vol. XVII, No. 1, 1975.
- (20) Oren, T.I., "Software Simulation of Combined Continuous and Discrete Systems", Simulation, Vol. 28, No. 2, pp. 33-45; Feb. 1977.
- (21) Cordella, L., M.J.B. Duff and S. Levialdi, "Comparing Sequential and Parallel Processing of Pictures", Proc. 3rd Joint Intl. Conf. on Pattern Recognition, Coronado, California, Nov. 1976.
- (22) Winograd, S., "On the Speed Gained in Parallel Methods", New Concepts and Technologies in Parallel Information Processing, Edited by E.R. Caianiello, NATO Advanced Studies Institute Series, Series E, Applied Sciences No. 9, Noordhoff International 1975.
- (23) Forsythe, G.E. and W.R. Wasow, "Finite-Difference Methods for Partial Differential Equations", J. Wiley and Sons, New York, (1960).
- (24) Collatz, L., "The Numerical Treatment of Differential Equations", (Third Ed.), Springer-Verlag, New York, (1966).

- (25) Miller, R.E., "A Comparison of some Theoretical Models of Parallel Computation", IEEE Trans. Electron. Comp., Vol. C-22, pp. 710-717; August 1973.

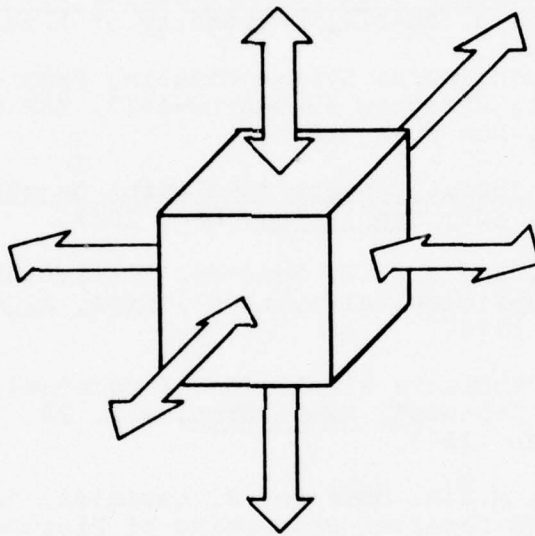


Fig. 1 Node processor diagram showing nearest six neighbor linkages.

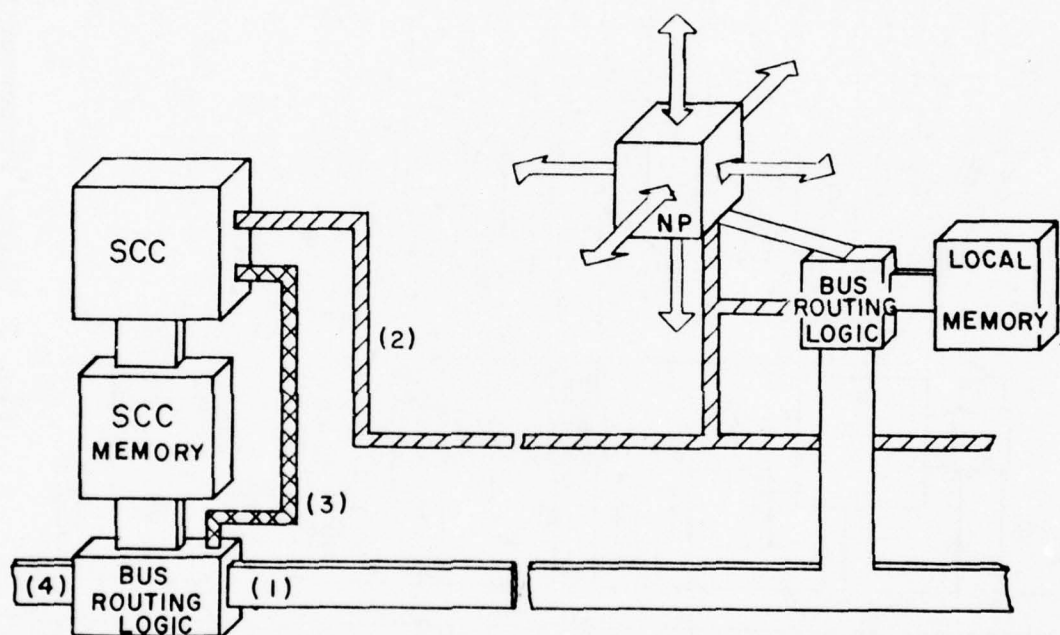


Fig. 2 Relationship of one node processor (NP) to the sector control computer (SCC) showing (1) the sector memory communication bus, (2) the sector control bus, (3) the sector routing control lines and (4) the external sector access bus.

BEST AVAILABLE COPY

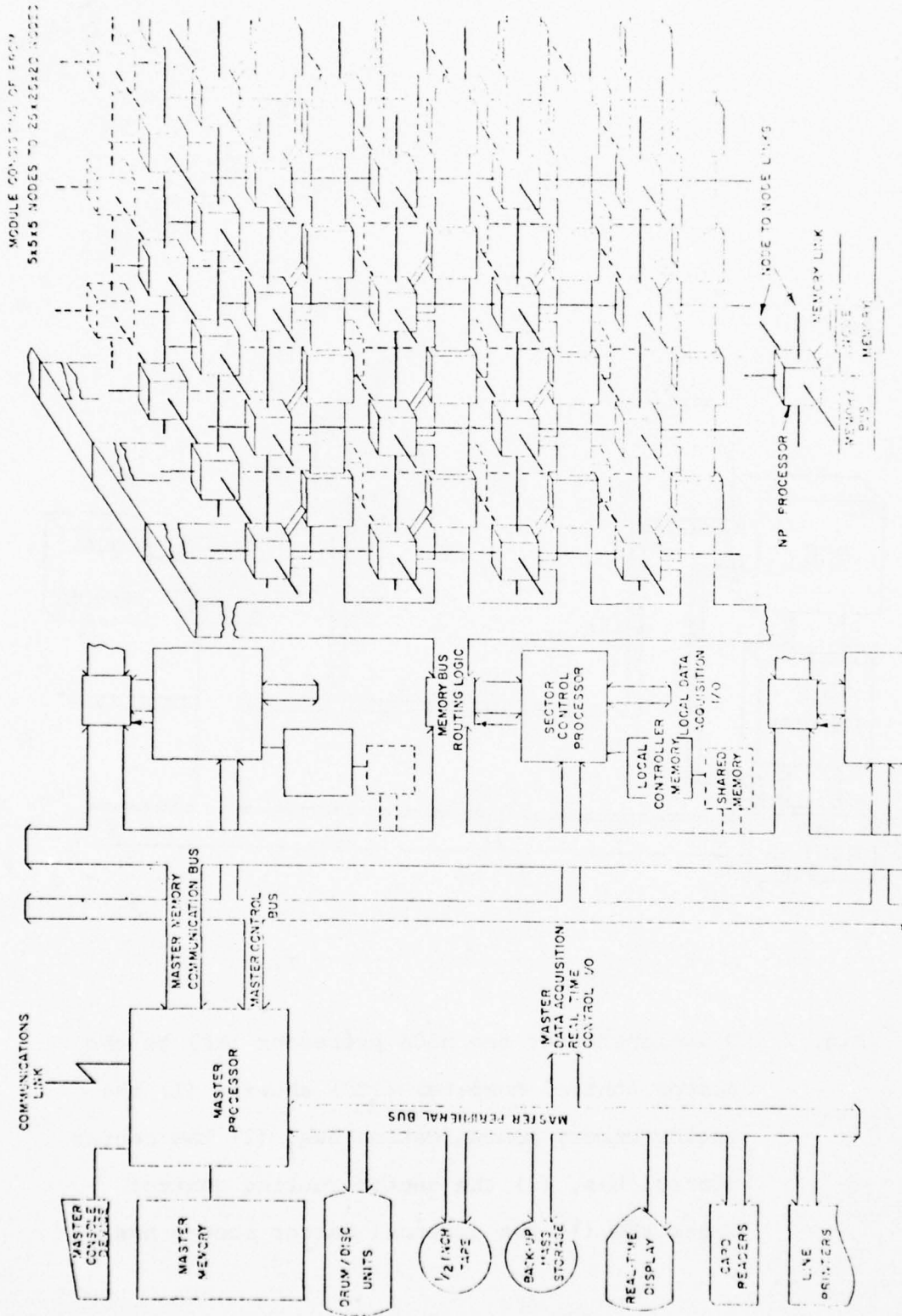


Fig. 3 Array Processor block diagram showing control computer interconnections.

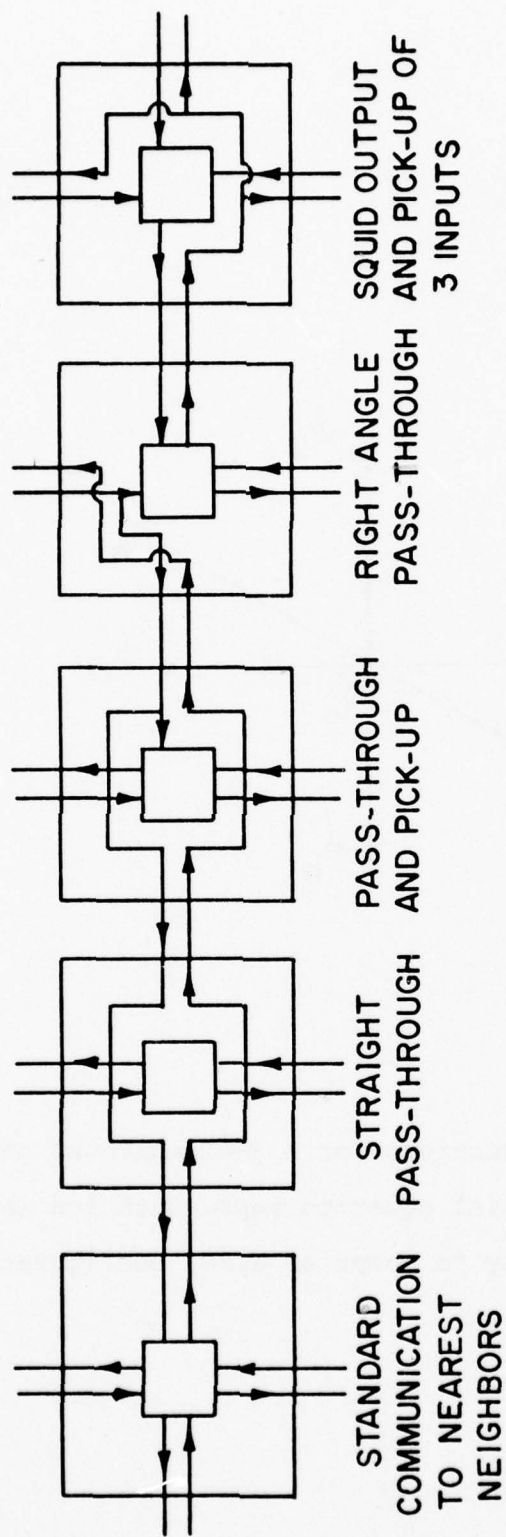


Fig. 4 TYPES OF COMMUTATION OPERATIONS

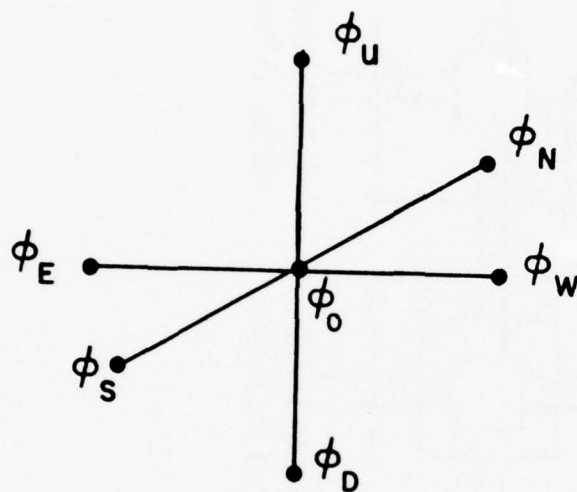
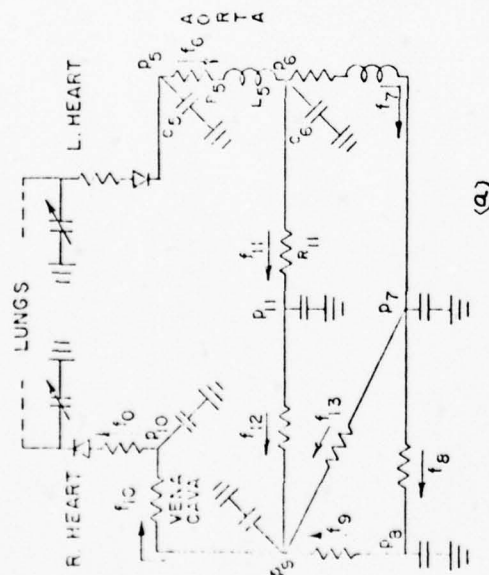
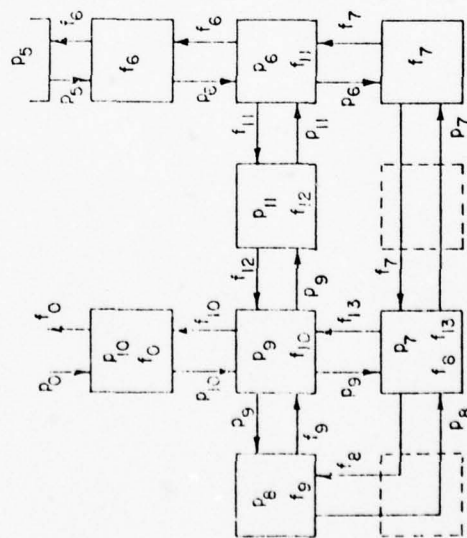


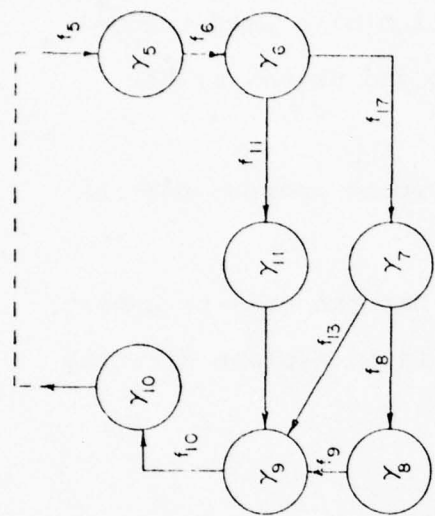
Fig. 5 Node connections for a 3-dimensional partial differential equation representation (note similarity to computer array configuration).



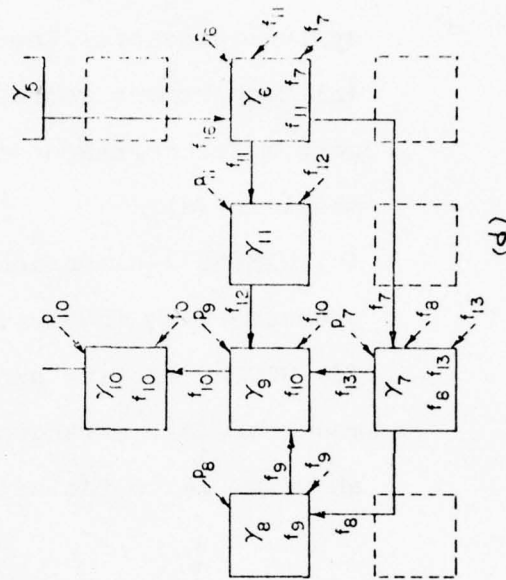
(a)



(c)



(b)



(d)

Fig. 6 (Over for captions)

Fig 6 (a) Circuit representation of a discretized model of the systemic part of the blood circulation system (pressure-flow dynamics).

 (b) Compartment representation of a mass-transport model corresponding to and driven by the model in (a).

 (c) WISPAC 2-dimensional routing program for the pressure-flow model of (a).

 (d) WISPAC routing program for the mass-transport model of (b), arranged to lie in a plane directly above or below the set-up in (c).

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

SOFTWARE FOR INTERVAL ARITHMETIC:
A REASONABLY PORTABLE PACKAGE

J. M. YOHE

Technical Summary Report #
March, 1977

ABSTRACT

We discuss the design and capabilities of a package of FORTRAN subroutines for performing interval arithmetic calculations. Apart from a relatively small number of primitives and constants, the package is directly transferrable to most large scale computers, and has been successfully implemented on IBM, CDC, and Honeywell equipment in addition to the UNIVAC 1110.

This package has been designed so as to be compatible with the AUGMENT precompiler, and includes interval analogs of appropriate standard FORTRAN operations and functions, as well as operations and functions peculiar to interval arithmetic. The result is that the user who has access to AUGMENT may write programs using interval arithmetic just as though FORTRAN recognized INTERVAL as a standard data type.

AMS(MOS) Subject Classification: 94-04

Key Words: Interval Arithmetic Program Package
Portable software

Work Unit No. 8 (Computer Science)

Sponsored by the United States Army under Contract Number DAAG29-75-C-0024 and a grant from the Waterways Experiment Station, Vicksburg, Mississippi.

SOFTWARE FOR INTERVAL ARITHMETIC:
A REASONABLY PORTABLE PACKAGE

J. M. YOHE

1. Introduction: One means of bounding the error in digital computation is through the use of interval, or range, arithmetic; instead of computing with approximate real numbers, one calculates with pairs of approximate real numbers -- the first member of a pair being a lower bound for the true result, and the second an upper bound. By this method, one can take into account such varied sources of error as uncertainty in input data, inaccuracies in mathematical formulae, and errors in approximation of real numbers and the operations on them. The theory of interval arithmetic is developed extensively elsewhere [5]; we shall not treat it here.

The major obstacle to the use of interval arithmetic is the unavailability of software. INTERVAL is not a standard data type in any production language that we know of; preparation of a package of subprograms to handle interval data is a nontrivial task. Since the representation of and operations on interval data are necessarily rather heavily dependent upon the architecture of the host computer, a package developed for one system can not, in general, be moved intact to a different system.

This paper describes an interval arithmetic package for use with FORTRAN. The power of the AUGMENT precompiler [2, 3] is employed to render the major part of the package independent of specific data representations, and the package is so designed that the parts which are representation-dependent are concentrated in a relatively small number of modules, most

Sponsored by the United States Army under Contract Number DAAG29-75-C-0024 and a grant from the Waterways Experiment Station, Vicksburg, Mississippi.

of which are easily adapted to new environments.

Although this package was written for the UNIVAC 1110, it has also been implemented on IBM, DEC, Honeywell, and CDC equipment. No major problems have been reported in transporting the package to these host systems.

The information given in this paper is not intended to be exhaustive. The interested reader will find detailed information on all aspects of the package in the technical manual [9].

2. Design of the package:

The viewpoint taken was that of the end user. We sought to make the package *complete, accurate, convenient to use, fail-safe, and transportable.*

Completeness: All appropriate ANSI Standard Fortran [1] operations and functions were implemented, along with some (such as tangent, hyperbolic sine, and hyperbolic cosine) which are not ANSI Standard but are normally implemented in the FORTRAN language anyhow. Since interval numbers can be regarded in a natural sense as belonging to an extension of the real number system, most arithmetic operations and special functions are meaningful. In addition, there are a large number of functions peculiar to interval arithmetic (such as union and intersection of intervals, midpoint, and half-length) which were also included in the package. Finally, input/output routines and conversions between intervals and standard data types (where appropriate) were implemented. A list of the functions and operations is given in Table 2.1.

Accuracy: It is well known that error is inherent in digital computations, and that most computer architectures are less than optimal from this point of view. (Recently, there has been increased interest in developing more hospitable architecture; see, for example, Lang and Shriver [4] and Ris [6].) Moreover, it is extremely difficult, if not impossible, to obtain the information required for rigorous bounding of hardware operations. Since interval arithmetic tends to be pessimistic anyhow, we felt that the calculation of bounds through straightforward application of *a priori* estimates such as Wilkinson's [7] would lead to intolerable inaccuracy. In addition, vital information concerning such phenomena as exponent range

faults is generally not available in existing systems. Consequently, the package was designed to be based on a set of arithmetic primitives of the type described in [8].

The special functions pose a different problem. A straightforward application of interval arithmetic to the algorithms used to compute these functions will yield unacceptably wide intervals, due to the dependency problem [5]. We addressed this problem by employing higher precision functions, bounding the results on the basis of accuracy information provided by the software supplier. This must be regarded as being less than completely satisfactory, since available error information is often sketchy and may not be completely rigorous; however, the bounding procedure takes these disadvantages into account, and the results can be regarded as being valid with extremely high probability.

Like the arithmetic routines, input/output routines need to be written from the ground up. Conversion routines supplied with standard FORTRAN systems have no provisions for obtaining the required bounds; moreover, most of them are of unknown, if not dubious, accuracy.

Convenience: By itself, no collection of routines to perform non-standard arithmetic is really convenient to use. Each operation must be performed by a call on one of the subprograms in the package; this means that the user must parse every expression himself and write his program in what amounts to assembly language. The best that can be done in this setting is to minimize the inconvenience. To this end, we have kept the package as internally consistent as possible. All entry points to the package bear the prefix INT; routines used by the package itself are prefixed with INT or BPA, according to their level. Thus, by avoiding variable and subprogram names beginning with these prefixes, the user may be assured of avoiding conflicts.

Calling sequences for the routines in the package are consistent and concise. No information is transmitted which is not absolutely essential to the function being performed. Where the result of a function or operation is a standard data type, the routine is implemented as a function of that type; otherwise, the routine is implemented as a subroutine. In the former case, the arguments are simply the operands; in the latter case, the arguments are the operands together with the result. (In the interest of flexibility, the endpoints of an interval are regarded as being a nonstandard data type.)

Convenience of use of any nonstandard data type is increased dramatically by the use of an appropriate precompiler. This package is specifically designed to be used with the AUGMENT precompiler, which allows the source FORTRAN code to be written as though FORTRAN recognized INTERVAL as a standard data type. In this case, just as above, the user must avoid conflicts with the package; although the source code will not contain references to the routines of the package, the output from AUGMENT, of course, will. In addition, the user must also avoid the function names and operators shown in the table, since these become reserved words in the extension of FORTRAN. In most cases, this should not be an onerous task.

Fail-safe: Errors can occur in many of the operations of the interval package, just as they can in REAL operations. It is our viewpoint that errors should not be ignored. Each subprogram in which an error can occur will call the error-handling routine, INTRAP, prior to returning control to the calling program. If no error has occurred, INTRAP simply returns control to the routine which called it. Otherwise, INTRAP takes the action specified by a table which resides in a COMMON block; the response depends on the error which has occurred, but usually includes a print-out which gives the user complete information on the error. The user may, if he chooses, alter

the response by changing the table.

Transportability: Transportability and flexibility of representation are closely linked. The package is based on three data types: BPA (mnemonic for Best Possible Answer), which is the data type of the endpoints of intervals, but is otherwise undefined except in a few primitive routines; INTERVAL, which is defined to be a BPA array of length 2; and EXTENDED, which is the data type in which evaluations of special functions are performed. In the UNIVAC version of the package, the representation of BPA is the same as that of REAL, and EXTENDED is a synonym for DOUBLE PRECISION.

The AUGMENT precompiler is used to extend the representations of these nonstandard data types throughout the package. The output of the AUGMENT precompiler is a set of routines which, apart from the arithmetic primitives which are written in assembly language, conforms as closely as possible to ANSI Standard FORTRAN.

There are less than twenty program modules which depend on the representations of BPA and EXTENDED numbers; many of these will need no alteration for most applications. Adaptation of the package to other hardware is discussed more fully in Section 4.

3. Use of the INTERVAL package:

If used as a collection of subroutines, without the benefit of the AUGMENT precompiler, the INTERVAL package is, of course, used just as any package of subprograms would be used. That is, the user must decide which routines must be invoked and in what order. We prefer to regard the INTERVAL package as an extension of the capabilities of the host computer system, and the AUGMENT precompiler as an instrument for extending FORTRAN to take advantage of the additional power. Thus we will address the question of use of the package in this context; the user who by reason of preference or necessity does not use the precompiler will have no difficulty in adapting this discussion to his needs.

Type declarations for INTERVAL variables: If X, Y, and Z represent INTERVAL variables, they must be declared as such by the statement

INTERVAL X, Y, Z

INTERVAL variables may be dimensioned; the only restriction is that if the FORTRAN compiler limits the number of dimensions of an array, that limit must be decreased by 1 for INTERVAL variables. The reason for this is that AUGMENT will declare INTERVAL variables as arrays.

Assignment of values to INTERVAL variables: Most real numbers can not be represented exactly in the computer. The error inherent in a statement such as

X = .1

may not be immediately obvious. If X is an INTERVAL variable, the above statement will assign a value to X, but that value will not, in general, be an interval containing the real number .1. In order to set X to an interval which does contain .1, one may write

X = '(.1, .1)\$', or X = 9H(.1, .1)\$ if the host compiler

does not accept quoted Hollerith literals. If the host compiler generates a sentinel for a Hollerith literal, and if the UNPACK primitive recognizes that sentinel, the terminal \$ may be omitted. Any string that is legal input for the formatted read (see discussion below and Appendix 1) is also acceptable to the routine which performs this conversion. Thus, on the UNIVAC 1110, the statement

X = '.1'

would also have the desired effect.

Reading INTERVAL variables: Two options are available in this package: a free format read and a formatted read.

The free format read will obtain the next data field from the input stream on the specified unit, convert it, and store the result in the specified INTERVAL variable. The calling sequence is

CALL INTRDF(UNIT, X)

The basic package will recognize units 5 (standard input) and 0 (reread), but the user may add other units or change unit designations as desired; this is discussed in the technical documentation. A data field may be any legal representation of an interval variable (see Appendix 1); however, for simplicity, one may be assured that the format (*number*, *number*), where *number* is any legal FORTRAN string representing an integer, fixed point number, or floating point number, is always valid. Embedded blanks between matching parentheses are *always ignored*. Fields may be separated by blanks (as many as desired), although if intervals are enclosed in parentheses as indicated above, blanks are unnecessary. Fields may be continued across card boundaries. The input stream remains uninterrupted so long as all reading is done by INTRDF and the unit number does not change. Once the input stream has been interrupted, INTRDF begins a new input stream with a new record.

The formatted read, as its name implies, reads interval data according to a specified format. This routine reads a vector of values (which may be of length 1). The calling sequence is

```
CALL INTRD(UNIT, FMT, A, N)
```

Unit is as in the free format read; A is the first location of the vector into which the data is to be read; and N is the length of the vector. FMT is an array of length 3; FMT(1) is the number of data items per record, FMT(2) is the number of characters to be ignored before each data field, and FMT(3) is the width of each data field. Note that these values are constant for each call to INTRD. A data field may be any legal representation of an interval variable; parentheses are optional, and embedded blanks are permitted. No other information is permitted within a data field.

Computing with INTERVAL variables: Expressions involving INTERVAL variables are written in standard FORTRAN syntax, just as though INTERVAL were a standard FORTRAN data type. A list of the operations and functions available in this package may be found in Appendix 2.

Mixed mode expressions are permitted, but their use is discouraged due to the high probability of introducing hidden error. For example, the expression

$$Y = 0.1 * X$$

where X and Y are INTERVAL variables, will not yield a correct value of Y; 0.1 will first be converted to REAL by the compiler, and AUGMENT will then cause that REAL number to be converted to a degenerate interval not containing .1. Multiplication will then occur using this erroneous interval.

Other operators and functions peculiar to interval arithmetic are implemented; examples include the intersection of two intervals, the union of two intervals, derivation of the midpoint and half-length, etc. These are listed in Appendix 2. Relational operators are also implemented, but

they take on different meanings in the context of interval arithmetic; see Appendix 2 for details.

Writing INTERVAL variables: The write routine will convert a vector (possibly of length 1) of INTERVAL variables to external format and write it on the specified output unit according to the given format. The external representation of each interval is guaranteed to contain the interval, and is the smallest interval representable in the given format which does so. The calling sequence is

```
CALL INTWR(UNIT, FMT, A, N)
```

The basic package will recognize units 6 (standard printer) and 1 (standard punch), but again the user may change designations and/or add units at will. If an illegal output unit is specified, INTWR will use the standard printer instead.

FMT is now an integer array of length 4. The first three values are the same as for INTRD (except that ignored characters in the output record are filled with blanks); FMT(4) is a carriage control character for use where appropriate. This character must be either '0' or ' ', denoting double spacing or single spacing, respectively. The width of each data field specified by the format must be at least great enough to permit the package to convert one significant digit; in the 1110 version, this is 15 characters, assuming a 2-digit exponent. Add 2 characters for each additional exponent digit in the external format. If an illegal format is specified, the routine will default to a standard format.

A and N are as in the formatted read.

Errors: The package is designed to detect all errors as they occur. The user may elect any of the available responses for any possible error (See Appendix 3); however, the default response is to print an error message and halt the computation except in those cases where viable alternatives

exist. Those cases are few indeed; they comprise arithmetic underflows (where the offending value is set either to zero or to the properly-signed number of smallest magnitude, as appropriate) and errors occurring on output (where the write routine uses standard modes of output rather than electing to scrub the computation and lose the output altogether). In the former case, the computation proceeds without notice to the user; in the latter case, a message is printed after the output is complete. The method of changing the default responses to errors is discussed in the technical documentation.

Producing an object program: Unless a sophisticated job control language allows for an automatic (from the user's point of view) invocation of the AUGMENT precompiler and the FORTRAN compiler, the generation of an object program is a two-step procedure:

1. Use AUGMENT to translate the source program into a FORTRAN program compatible with the compiler. This can be accomplished with a run stream of the following type:

```
invoke AUGMENT
description decks for BPA and INTERVAL (supplied with the package)
*BEGIN
source program
*END
```

AUGMENT will write the translated program on Unit 20.

2. Compile the output of AUGMENT using the standard FORTRAN compiler and execute the resulting program in the usual manner. The user must insure that the BLOCK DATA modules are included when the program is processed by the linkage editor.

4. Adaptation of the package:

Adaptation of the package to other hardware is not difficult provided one has access to the AUGMENT precompiler. The necessary steps are:

1. Decide on data representations for the interval endpoints and for EXTENDED precision numbers.
2. Code or revise primitives, as necessary.
3. Process the package through the AUGMENT precompiler and compile the resulting FORTRAN code.
4. Check the package.
5. Tune and recheck the package.

We discuss each of these steps in greater detail.

Data representations: Normally, the representation for interval endpoints will be the same as REAL and EXTENDED will be the same as double precision. These choices will simplify the adaptation of the package; however, for special purposes such as higher precision interval arithmetic, other choices may be made. There are several implicit assumptions which will, to a certain extent, govern the choices of representations:

a. The portion of the package which performs endpoint evaluations (known as type BPA) will contain explicit routines to perform all operations. As designed, it is assumed that conversion from BPA to REAL is exact, although conversion in the other direction need not be. This is done to facilitate adaptation to two's complement hardware, where the negative of a real number is not necessarily representable; we assume that the negative of every BPA number is representable.

b. It is assumed that EXTENDED is bound to a higher precision than is BPA. Moreover, we assume that every BPA number and every FORTRAN integer can be represented exactly in EXTENDED format. For the evaluation of special functions, we assume that a complete supporting package exists

for type EXTENDED, and that bounds on the accuracy of these routines are available.

Primitives: There are nineteen primitives which depend on the representation of BPA and EXTENDED numbers in the host system. Two of these are BLOCK DATA modules, which contain various representation dependent constants; eight are written in FORTRAN and depend only on BPA format being the same as REAL and EXTENDED being the same as DOUBLE PRECISION; three depend on both data representations and the (nonstandard) FLD function; one contains FORMAT statements which may be representation dependent; and five are arithmetic primitives which must necessarily be recoded for any change in data representation. The arithmetic primitives are, in fact, written in assembly language.

In addition, INTRD and INTRDF, while not technically primitives, contain nonstandard READ statements which recognize the END OF FILE condition. If the host compiler does not recognize this form of READ statement, those statements will need to be modified.

Complete documentation of these primitives is given in the technical manual. It does not seem appropriate to go into greater detail here.

AUGMENT processing: The use of the AUGMENT precompiler preserves both naturality of expression and flexibility. Most of the INTERVAL package is written in terms of the nonstandard types BPA, EXTENDED, and INTERVAL. The binding to specific data representations is accomplished through the primitives, and these bindings are extended through the remainder of the package by the use of AUGMENT. Every effort has been made to write the package so that the output of the AUGMENT precompiler will be ANSI Standard FORTRAN. There is no requirement that AUGMENT be available on the target computer; the pre-processing can just as well be done on any computer, with the resulting

FORTRAN code being brought to the target system for compilation.

Checking the package: A collection of test programs is provided with the INTERVAL package. Successful execution of these programs is reasonably good assurance that the primitives have been implemented properly.

Tuning the package: The price paid for the degree of flexibility present in the source code for this package is quite likely to be decreased efficiency in the object code. For example, since the format of BPA numbers is arbitrary, conversion from REAL to BPA will generate a call on a subprogram which is responsible for performing this task (this subprogram is, of course, a primitive). If BPA numbers are the same as REAL, this will result in unnecessary overhead; an in-line replacement operation would perform the same task at considerably less cost. AUGMENT can not be instructed to make this modification; thus, for greatest efficiency, it will be necessary to examine the output of AUGMENT and replace calls of this type by in-line replacement statements. There are, of course, many other possibilities, depending on representation; for example, if the hardware has double precision capability, one could change calls on the interval replacement subroutine to in-line replacement statements using the double precision hardware.

A certain amount of care must be exercised in tuning the package. For example, the routines which evaluate BPA relational operators call on the BPA subtract routine. This should not be altered unless the hardware subtract always produces a result of the same sign as the true result, even in cases of underflow and overflow. If the hardware sets an underflow to zero, or gives garbage when overflow occurs, then the hardware subtract must not be used.

Needless to say, the package must be rechecked whenever any changes are made.

5. Conclusion:

In this paper, we have sketched the design and use of a package for performing calculations in interval arithmetic. The package is both flexible and transportable; adaptation of the package to other systems can be accomplished by rewriting a maximum of nineteen primitive modules, most of which are easily adapted to a new host system. Further details of the package are provided in the technical documentation.

REFERENCES

1. ANSI Standard FORTRAN, American National Standards Institute, New York, 1966.
2. Crary, F. D. The AUGMENT precompiler I. User information. The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report # 1469, December, 1974.
3. _____. The AUGMENT precompiler II. Technical documentation. The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report # 1470,
4. Lang, Allan L. and Shriver, Bruce D. The design of a polymorphic arithmetic unit. *Third IEEE - TCCA Symposium on Computer Arithmetic*, November, 1975, 48 - 55.
5. Moore, Ramon E. *Interval Analysis*. Prentice - Hall, Inc., Englewood Cliffs, N. J., 1966
6. Ris, Frederic N. A unified decimal floating-point architecture for the support of high-level languages (extended abstract). *SIGNUM Newsletter* 11, 3 (October, 1976), 18 - 22.
7. Wilkinson, J. H. Rounding errors in algebraic processes. *Notes on Applied Science No. 32*, Her Majesty's Stationery Office, London, 1963.
8. Yohe, J. M. Roundings in floating-point arithmetic. *IEEE Trans. Computers C-22* (1973), 577 - 586.
9. _____. The INTERVAL Arithmetic package. The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report (forthcoming).

APPENDIX 1

STANDARD FORTRAN NUMBER AND INTERVAL NUMBER REPRESENTATIONS

DIGIT	::= 0 1 2 3 4 5 6 7 8 9
SIGN	::= + -
INTEGER	::= NULL <SIGN> <INTEGER><DIGIT>
RADIX	::= .
FIXEDPOINT	::= <INTEGER><RADIX> <FIXEDPOINT><DIGIT>
EXPSEP	::= E D
EXPONENT	::= <SIGN> <EXPSEP> <EXPSEP><SIGN> <EXPONENT><DIGIT>
NUMBER	::= <INTEGER> <FIXEDPOINT> <INTEGER><EXPONENT> <FIXEDPOINT><EXPONENT>
ENDPTSEP	::=
COMMA	::= ,
INTERVAL	::= <NUMBER> (<NUMBER>) <NUMBER><ENDPTSEP><NUMBER> (<NUMBER><ENDPTSEP><NUMBER>) (<NUMBER><COMMA><NUMBER>)

INTERVAL INPUT RULES

FORMATTED INPUT:

One and only one <INTERVAL> shall appear in any one field.

Embedded blanks are permitted; they will be ignored.

FREE FORMAT INPUT:

Leading blanks are always ignored.

Blanks within matching pairs of parentheses are always ignored.

Commas within matching pairs of parentheses are regarded as endpoint separators.

A field consists of exactly one <INTERVAL>.

A field is terminated by

1. A visible blank;
2. Any of the characters '\$', '#', '=';
3. A comma occurring outside of a matching pair of parentheses;
4. Any nonblank character following a matching right parenthesis (If such character is not '\$', '#', '=', or ',', it will be regarded as the first character of the next field);
5. A left parenthesis or colon occurring outside of a matching pair of parentheses. (Such character will be regarded as the first character of the next field).

If a left parenthesis is encountered, the scan proceeds to the matching right parenthesis regardless of what characters are encountered, except that '\$', '#', and '=' always terminate the field.

ALL INPUT:

A null field is taken to represent the interval (0, 0).

A field containing <NUMBER> or (<NUMBER>) is taken to represent a degenerate interval; this number is converted and rounded down for the left endpoint, and up for the right endpoint.

If a field contains two <NUMBER>s, the first will be converted and rounded down for the left endpoint, and the second will be converted and rounded up for the right endpoint.

APPENDIX 2

OPERATION	DEFINITION/EXPLANATION	TYPE	VIA AUGMENT	INVOCATION	ROUTINE TYPE
ARITHMETIC					
Add	Sum of two intervals	X	XA + XB	INTADD(XA, XB, XR)	S
Subtract	Difference of two intervals	X	XA - XB	INTSUB(XA, XB, XR)	S
Multiply	Product of two intervals	X	XA * XB	INTMUL(XA, XB, XR)	S
Divide	Quotient of two intervals	X	XA / XB	INTDIV(XA, XB, XR)	S
EXPONENTIATION					
to BPA	Raise interval to BPA power	X	XA ** BB	INTXXB(XA, BB, XR)	S
to EXTENDED	Raise interval to EXTENDED power	X	XA ** EB	INTXXE(XA, EB, XR)	S
to INTEGER	Raise interval to INTEGER power	X	XA ** IB	INTXXI(XA, IB, XR)	S
to INTERVAL	Raise interval to INTERVAL power	X	XA ** XB	INTXXX(XA, XB, XR)	S
MATHEMATICAL					
Absolute value	$(x : x \in XA)$	X	ABS(XA)	INTABS(XA, XR)	S
Arc cosine	Arc cosine of interval XA	X	ACOS(XA)	INTACS(XA, XR)	S
Arc sine	Arc sine of interval XA	X	ASIN(XA)	INTASN(XA, XR)	S
Arc tan (2 args)	Arc tangent of XA / XB	X	ATAN2(XA, XB)	INTAT2(XA, XB, XR)	S
Arc tangent	Arc tangent of interval XA	X	ATAN(XA)	INTATN(XA, XR)	S
Cube root	Cube root of interval XA	X	CBRT(XA)	INTCBT(XA, XR)	S
Cosine	Cosine of interval XA	X	COS(XA)	INTCOS(XA, XR)	S
Hyperbolic cosine	Hyperbolic cosine of interval XA	X	COSH(XA)	INTCSH(XA, XR)	S
Exponential	e^x	X	EXP(XA)	INTEXP(XA, XR)	S
Integer	Smallest interval with integer endpoints containing the interval XA	X	INT(XA)	INTINT(XA, XR)	S
Natural logarithm	Log to the base e of interval XA	X	LN(XA) or LOG(XA)	INTLN(XA, XR)	S
Common logarithm	Log to the base 10 of interval XA	X	LOG10(XA)	INTLOG(XA, XR)	S
Sine	Sine of interval XA	X	SIN(XA)	INTSIN(XA, XR)	S
Hyperbolic sine	Hyperbolic sine of interval XA	X	SINH(XA)	INTSNH(XA, XR)	S
Square root	Square root of interval XA	X	SQRT(XA)	INTSQRT(XA, XR)	S
Tangent	Tangent of interval XA	X	TAN(XA)	INTTAN(XA, XR)	S
Hyperbolic tangent	Hyperbolic tangent of interval XA	X	TANH(XA)	INTTNH(XA, XR)	S
FIELD					
Infimum	Left endpoint of interval XA	B	INF(XA)	INTINL(BA, XR) (insertion)	S
Supremum	Right endpoint of interval XA	B	SUP(XA)	INTINF(XA) (extraction)	B
				INTSPL(BA, XR) (insertion)	S
				INTSUP(XA) (extraction)	B
SPECIAL INTERVAL FUNCTIONS					
Compose	Form interval from two BPA endpoints	X	COMPOS(BA, BB)	INTCPS(BA, BB, XR)	S
Distance	$\text{Max}(\text{Inf}(XA) - \text{Inf}(XB) , \text{Sup}(XA) - \text{Sup}(XB))$	B	DIST(XA, XB)	INTDST(XA, XB, BR)	S
Half length	$(\text{Sup}(XA) - \text{Inf}(XA)) / 2$, rounded up	B	HLGTH(XA)	INTHLB(XA, BR)	S
Length	$\text{Sup}(XA) - \text{Inf}(XA)$, rounded up	B	LGTH(XA)	INTLGB(XA, BR)	S

OPERATION	DEFINITION/EXPLANATION	RESULT TYPE	ROUTINE INVOCATION	ROUTINE TYPE
SPECIAL INTERVAL FUNCTIONS (continued)				
Magnitude	$\text{Sup}(\text{Abs}(XA))$	B	MAG(XA)	INTMAG(XA, BR) S
Midpoint	$(\text{Sup}(XA) + \text{Inf}(XA)) / 2$, rounded nearest	B	MDPT(XA)	INTMDB(XA, BR) S
Magnitude	$\text{Inf}(\text{Abs}(XA))$	B	MIG(XA)	INTMIG(XA, BR) S
Pivot	$\sqrt{\text{Mag}(XA) \times \text{Mig}(XA)}$, rounded down	B	PIVL(XA)	INTPVL(XA, BR) S
	same, rounded up	B	-	INTPU(XA, IB, BR) S
Intersection	Set-theoretic intersection of XA and XB	B	PIVU(XA)	INTPU(XA, BR) S
Sign	+1 if $\text{Inf}(XA) > 0$, -1 if $\text{Sup}(XA) < 0$, 0 if $0 \in XA$	X	XA.INTSCT.XB	INTSCT(XA, XB, XR) S
Size	$(\text{Abs}(\text{Inf}(XA)) + \text{Abs}(\text{Sup}(XA))) / 2$	I	SGN(XA)	INTSGN(XA) I
Union	Smallest interval containing both XA, XB	B	SIZE(XA)	INTSIX(XA, BR) S
CONVERSION				
PH \rightarrow X	Packed Hollerith to Interval	X	CTX(string)	INTASG(HA, XR) S
E \rightarrow X	Extended to Interval, bounded at IBth dig.	X	-	INTBND(EA, IB, XR) S
B \rightarrow X	BPA to Interval	X	CTX(BA)	INTCBX(BA, XR) S
E \rightarrow X	Extended to Interval	X	CTX(EA)	INTCEX(EA, XR) S
UH \rightarrow X	Unpacked Hollerith to Interval (width IB)	X	-	INTCHX(HA, IB, XR) S
I \rightarrow X	Integer to Interval	X	CTX(IA)	INTCIX(IA, XR) S
R \rightarrow X	Real to Interval	X	CTX(RA)	INTCRX(RA, XR) S
X \rightarrow B	Interval to BPA	B	CTB(XA)	INTCXB(XA, BR) S
X \rightarrow E	Interval to Extended	E	CTE(XA)	INTCXE(XA, ER) S
X \rightarrow UH	Interval to unpacked Hollerith (width IB)	UH	-	INTCXH(XA, HR, IB) S
X \rightarrow I	Interval to Integer	I	CTI(XA)	INTCXI(XA, IR) I
X \rightarrow R	Interval to Real	R	CTR(XA)	INTCXR(XA, RR) R
SERVICE				
Store	Replacement operator	X	XR = XA	INTSTR(XA, XR) S
Negate	Unary minus	X	-XA	INTNEG(XA, XR) S
LOGICAL AND RELATIONAL				
Bad interval	$\text{Inf}(XA) > \text{Sup}(XA)$	L	BAD(XA)	INTBAD(XA) L
Element of	$BA \in XB$	L	BA.E. XB	INTELE(BA, XB) L
Good interval	$\text{Inf}(XA) \leq \text{Sup}(XA)$	L	OK(XA)	INTOK(XA) L
Subset of	XA contained in XB	L	XA.SUBSET.XB	INTSUBS(XA, XB) L
Set-equal	$\text{Inf}(XA) = \text{Inf}(XB)$ and $\text{Sup}(XA) = \text{Sup}(XB)$	L	XA.SEQ. XB	INTSEQ(XA, XB) L
Set-greater-equal	$\text{Sup}(XA) \geq \text{Sup}(XB)$	L	XA.SGE. XB	INTSGE(XA, XB) L
Set-greater	$\text{Sup}(XA) > \text{Sup}(XB)$	L	XA.SGT. XB	INTSGT(XA, XB) L
Set-less-equal	$\text{Inf}(XA) \leq \text{Inf}(XB)$	L	XA.SLE. XB	INTSLE(XA, XB) L
Set-less	$\text{Inf}(XA) < \text{Inf}(XB)$	L	XA.SLT. XB	INTSLT(XA, XB) L
Set-not-equal	$\text{Inf}(XA) \neq \text{Inf}(XB)$ or $\text{Sup}(XA) \neq \text{Sup}(XB)$	L	XA.SNE. XB	INTSNE(XA, XB) L

OPERATION	DEFINITION/EXPLANATION	RESULT ROUTINE INVOCATION TYPE VIA AUGMENT DIRECT	ROUTINE TYPE
LOGICAL AND RELATIONAL(continued)			
Superset	XA contains XB	L XA.SPRSET, XB	L
Value-equal	Inf(XA) = Sup(XB) = Inf(XB) = Sup(XB)	L XA.VEQ. XB	L
Value-greater-eq.	Inf(XA) > Sup(XB)	L XA.VGE. XB	L
Value-greater	Inf(XA) > Sup(XB)	L XA.VGT. XB	L
Value-less-equal	Sup(XA) < Inf(XB)	L XA.VLE. XB	L
Value-less	Sup(XA) < Inf(XB)	L XA.VLT. XB	L
Value-not-equal	XA does not intersect XB	L XA.VNE. XB	L
INPUT/OUTPUT			
Read	Read interval vector, formatted	X -	S
Read, free format	Read next interval from data stream	X -	S
Write	Write interval vector, formatted	- -	S
MISCELLANEOUS			
Reduce	Reduce argument of trig function to principal range	X -	S
Unpack	Unpack packed Hollerith	H -	S
ERROR HANDLING			
Trap	Detect errors in interval package (arguments in COMMON)	- -	S

NOTES ON TABLE:

DATA TYPES: B = BPA; E = EXTENDED; I = INTEGER; L = LOGICAL; R = REAL; X = INTERVAL; H = HOLLERITH; UN = UNPACKED HOLLERITH; PH = PACKED HOLLERITH; D = DOUBLE PRECISION

ROUTINE TYPES: S = SUBROUTINE; any other letter denotes a function of the indicated type (see above).

VARIABLE NAMES: The first letter indicates the type of the variable. The second letter is R for RESULT, A or B for argument; other letters may be used for special meanings.

APPENDIX 3

FAULT NUMBER	MEANING	ACTION CODE
0	No fault	0
1	Left endpoint - no fault	4*
2	Right endpoint - overflow	3
3	no fault	0
4	infinity	4*
5	underflow	4*
6	overflow	3
7	overflow	4*
8	infinity	3
9	infinity	3
10	infinity	3
11	infinity	3
12	underflow	0
13	underflow	4*
14	underflow	3
15	underflow	0
16	Division by zero	3
17	Zero raised to the zero power	1
18	Square root of a negative number	3
19	Logarithm of a nonpositive number	3
20	Underflow during computation of a BPA result	0
21	Overflow during computation of a BPA result	3
22	Intersection of disjoint intervals	3
23	Arc cosine or arc sine argument out of range	3
24	Inverted interval	4
25	Illegal input character	4
26	Illegal input format specification	4
27	Illegal output format specification	1
28	Input string too long	4
29	Illegal or unspecified input unit	4
30	End of file on input unit	1
31	Illegal or unspecified output unit	1
32	Conversion array overflow during base conversion	4†
33	Unrecognized error	4

* Denotes that the fault is logically impossible

† This action should not be changed, since any other action could result in a recursive call on INTRAP from INTCXH.

In the event that a fault occurs, the corresponding action code governs the response of the INTRAP routine. The action codes, and their responses, are:

- 0 Return to the calling program without taking any action
- 1 Print error message and return to the calling program
- 2 Print error message, trace call sequence, and return
- 3 Print error message, trace call sequence, step error counter in Executive program, and return
- 4 Print error message, trace call sequence, and halt computation

AUTOMATIC DIFFERENTIATION OF COMPUTER PROGRAMS

Gershon Kedem

University of Wisconsin - Madison
Mathematics Research Center

ABSTRACT

A method for the automatic differentiation of computer functions (subroutines) written in a high level language is discussed.

A theory is developed to show that most functions that arise in applications can be differentiated automatically. It is shown how one can take a FORTRAN function (subroutine) and, with the aid of a precompiler, obtain a FORTRAN subroutine that computes the original function and its desired derivatives.

Implementation of two types of differentiation is described:

- 1) Automatic Taylor series expansion of FORTRAN programs.
- 2) Automatic Gradient calculation of FORTRAN functions.

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024

AUTOMATIC DIFFERENTIATION OF COMPUTER PROGRAMS

Gershon Kedem

1. Introduction:

The use of recurrence relations to compute derivatives is not a new idea. We can trace this idea back as far as 1932 when it was used to compute the Emden functions by means of Taylor series expansion by J. R. Airy (See [1]). This idea has apparently been rediscovered many times. In 1964 R. E. Moore [7] showed how one could automatically get Taylor series expansions of FORTRAN-like expressions to solve initial value problems. See [1, 2, 5, 7, 8, 10]. The automatic computation of partial derivatives was implemented in 1967 by A. Reiter and J. Gray [11, 14] and later by J. Wertz [12], D. Kuba and L. B. Rall [13], and R. E. Pugh [9]. These are programs known to us but the list is probably not complete.

This paper suggests a way to extend the process to functions that can be written in an algebraic computer language (FORTRAN, ALGOL and so on) namely: piecewise factorable functions.

Let us look at computer programs for the evaluation of numerical functions that arise in applications.

We shall use the FORTRAN language but the following discussion applies to any other high level algebraic language.

We look at FORTRAN subroutines and functions that evaluate mathematical functions. We assume no I/O is involved and that "random numbers" are not used. All such procedures have a few features in common:

- 1) For every set of values of the formal arguments there is a fixed finite

sequence of instructions executed (provided that these values are within the domain of definition of the function and we use a correct procedure).

2) If we regard DO loops, logical statements and GOTO statements only as convenient tools for defining sequences of instructions, we see that all such sequences consist of arithmetic operations and calls to library functions.

3) At each step one only uses previously defined values.

4) Most of the functions computed are piecewise differentiable or actually piecewise analytic.

We now define a mathematical model for such functions. We will use subscripts to denote sequences and superscripts to denote components of vectors.

2. Definitions:

i) Let \mathcal{L} be a finite set of real functions of one or more real arguments including the identity function. We call \mathcal{L} the set of basic library functions.

Denote by $\pi_i^n: \mathbb{R}^n \rightarrow \mathbb{R}$ the projection of the i th coordinate, that is

$$\pi_i^n(x^1, \dots, x^n) = x^i.$$

ii) A function $f: D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ is a factorable function if there exists a finite sequence of functions f_1, \dots, f_k that satisfy the following:

a) $\forall 1 \leq j \leq k \quad f_j: D \rightarrow \mathbb{R}.$

b) $f = f_k$, the last term in the sequence.

c) $f_1 = \pi_1^n, f_2 = \pi_2^n, \dots, f_n = \pi_n^n.$

d) for $n < j \leq k$ f_j is either a composition of a basic library function with one or more functions that appeared earlier in the sequence or f_j is identically constant.

iii) We say that the sequence f_1, \dots, f_k represents f .

iv) We call the sequence f_1, \dots, f_k a Basic Representation of f .

Example 1:

$$\mathcal{L} = \{+, -, *, /, \text{Sin}, \text{Cos}, \text{Exp}, \dots\}.$$

$$f(x) = \text{Cos}(x) \text{Exp}(\text{Sin}(x)) + \text{Sin}^2(x).$$

A Basic Representation of f :

$$f_1 = \pi_1^n$$

$$f_2 = \text{Cos}(f_1)$$

$$f_3 = \text{Sin}(f_1)$$

$$f_4 = \text{Exp}(f_3)$$

$$f_5 = f_3 * f_3$$

$$f_6 = f_2 * f_4$$

$$f_7 = f_5 + f_6.$$

Definition:

$f: D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^s$ is factorable if f^j $1 \leq j \leq s$ are factorable.

Remarks:

1) Only if all compositions are well defined can we call f factorable.

2) A basic representation of a function is not unique.

As an immediate consequence of the definitions we have

Proposition 1:

If all the basic library functions in the representation of f are: continuous, C^k , analytic, ... so is f .

Proposition 2:

The composition of two factorable functions is a factorable function.

The following lemma is somewhat trivial but it was brought in order to introduce some notation.

Lemma:

Assume that the sequence of functions f_1, f_2, \dots, f_k satisfy the following:

- a) $f_i: D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^{s(i)}$, that is: all functions have the same domain.
- b) For each i $1 \leq i \leq k$ f_i is either a factorable function on D or f_i is a composition of a factorable function with one or more functions that appeared earlier in the sequence. Or f_i is identically constant.

Then for every $1 \leq j \leq k$ f_j is a factorable function.

Definition: We say that the sequence f_1, \dots, f_j represents f_j and we call the sequence f_1, \dots, f_j A Factorable Representation of f_j .

Definition:

A function $f: D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a piecewise factorable function if there exists countable number of sets U_1, U_2, \dots such that $D \subseteq \bigcup U_j$ and f restricted to each U_j is a factorable function.

Examples:

- 1) $\text{Min}(\cdot, \cdot), \text{Max}(\cdot, \cdot), \text{Abs}(\cdot)$.
- 2) Spline functions
- 3) $A \rightarrow A^{-1}$ (A $N \times N$ matrix).

Remarks:

In practice most computer programs written in high level languages like FORTRAN and ALGOL, compute (represent!) piecewise factorable functions. One can write representations of such functions simply by following the path of execution.

3. Differentiation of piecewise factorable functions:

We now turn to the question of computing derivatives of piecewise factorable functions. We start with an example.

Example 2:

We take \mathcal{L} and f to be as in example 1.

$$\mathcal{L} = \{+, -, *, /, \text{Sin}, \text{Cos}, \text{Exp}, \dots\}$$

$$f(x) = \text{Cos}(x)\text{Exp}(\text{Sin}(x)) + \text{Sin}^2(x).$$

A Basic Representation of f is:

$$f_1 = \pi_1^1$$

$$f_2 = \text{Sin}(f_1)$$

$$f_3 = \text{Exp}(f_2)$$

$$f_4 = \text{Cos}(f_1)$$

$$f_5 = f_2 * f_2$$

$$f_6 = f_4 * f_3$$

$$f_7 = f_6 + f_5$$

We now look at the following sequence of functions:

$$\begin{array}{ll} f_1 = \pi_1^2 & ; \quad \hat{f}_1 = \pi_2^2 \\ f_2 = \text{Sin}(f_1) & ; \quad \hat{f}_2 = \text{Cos}(f_1) * \hat{f}_1 \\ f_3 = \text{Exp}(f_2) & ; \quad \hat{f}_3 = f_3 * \hat{f}_2 \\ f_4 = \text{Cos}(f_1) & ; \quad \hat{f}_4 = -\text{Sin}(f_1) * \hat{f}_1 \\ f_5 = f_2 * f_2 & ; \quad \hat{f}_5 = f_2 * \hat{f}_2 + \hat{f}_2 * f_2 \\ f_6 = f_4 * f_3 & ; \quad \hat{f}_6 = f_4 * \hat{f}_3 + \hat{f}_4 * f_3 \\ f_7 = f_6 + f_5 & ; \quad \hat{f}_7 = \hat{f}_6 + \hat{f}_5 \end{array}$$

Please note:

a) The combined sequence (f_j, \hat{f}_j) is a factorable representation of a function from $R^2 \rightarrow R^2$.

b) The functions f_j are the same as before except now they are regarded as functions of two arguments.

c) There is a simple correspondence between f_j and \hat{f}_j , that is: \hat{f}_j only depends on what functions are composed in the j th stage.

d) It is not hard to see (or prove by induction) that $f_7(x_0, 1) = f(x_0)$ and $\hat{f}_7(x_0, 1) = \frac{d}{dx} f \Big|_{x=x_0}$, or in general if $x_0 = x(t_0)$, $y_0 = \frac{dx}{dt} \Big|_{t=t_0}$ then

$$f_7(x_0, y_0) = f(x(t_0)), \quad \hat{f}_7(x_0, y_0) = \frac{d}{dt} f(x(t)) \Big|_{t=t_0}$$

The above result is not a great surprise. All we have done was to systematically use the chain rule and the fact that we know the derivatives of the basic library functions.

Example 3:

Let $[X]_j$ denote $\frac{1}{j!} \frac{d^j X}{dt^j}$. If one knows $[X]_j, [Y]_j$ $j = 0, 1, \dots$ and $Z = X \pm Y$ then one can compute $[Z]_j$ by:

$$[Z]_j = [X]_j \pm [Y]_j \quad j = 0, 1, \dots$$

If $Z = X * Y$ then by Libniz rule

$$[Z]_j = \sum_{k=0}^j [X]_k * [Y]_{j-k} \quad j = 0, 1, \dots$$

If $Z = \text{Exp}(X)$ then

$$[Z]_j = \sum_{k=0}^{j-1} ((j-k)/j) * [Z]_k * [X]_{j-k} \quad j = 1, 2, \dots$$

It is well known that there are recurrssion relations that enable one to compute successive derivatives of functions that satisfy rational differential equations.

As a matter of fact, the discussions in [1] and [7] show that it is possible to write such a recursion relations for functions satisfying differential equations of the form $y' = f(t, y)$ where f is a factorable function and \mathcal{L} is a set containing the arithmetic operations and functions satisfying rational differential equations.

If we make systematic use of the chain rule and the recursion rules for computing successive derivatives of basic library functions, we could construct, using a basic representation of a function, a factorable representation of the original function and its successive derivatives.

Example 4:

Let us take \mathcal{L} to be as in Example 3. It is not hard to see how one could derive recursion relations that will enable one to compute partial derivatives.

The following is motivated by the above examples:

Let \mathcal{L} be a set of basic library functions and let T be an operator that maps functions to functions. Let us assume the following:

- a) T is defined for all factorable functions.
- b) If $f: D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ then $T[f]: E \subseteq \mathbb{R}^{n \cdot k} \rightarrow \mathbb{R}^k$ where $E = D \times \mathbb{R}^{(k-1) \cdot n}$.
- c) $T[(f^1, \dots, f^m)] = (T[f^1], \dots, T[f^m])$.
- d) For every $g \in \mathcal{L}$ $G = T[g]$ is a factorable function.
- e) For any $g \in \mathcal{L}$, $g: D \subseteq \mathbb{R}^s \rightarrow \mathbb{R}$ and for every $f: \tilde{D} \subseteq \mathbb{R}^n \rightarrow D$

$$T[g(f)] = T[g](T[f]).$$

- f) $T[c]$ (c a constant function) is a constant k -vector function and \exists a factorable function $C: \mathbb{R} \rightarrow \mathbb{R}^k$ such that $T[c] = C(c) \quad \forall c \in \mathbb{R}$.

Theorem:

Let \mathfrak{L} and T satisfy the above conditions. Let f be a factorable function $f: D \subseteq R^n \rightarrow R$ and let f_1, \dots, f_L be a basic representation of f , then $T[f]$ is a factorable function. Moreover if we replace the terms f_1, \dots, f_L by F_1, \dots, F_L where:

i) For $1 \leq i \leq n$ $F_i = \prod_i$ where \prod_i is a k -coordinate projection from $R^{n \cdot k} \rightarrow R^k$, that is:

$$\prod_i(X) = (X^{k(i-1)+1}, X^{k(i-1)+2}, \dots, X^{i \cdot k})$$

ii) For $n < i \leq L$

$$F_i = \begin{cases} G(F_{j_1}, \dots, F_{j_s}) & \text{if } f_i = g(f_{j_1}, \dots, f_{j_s}) \\ C(c) & \text{if } f_i \equiv c \end{cases}$$

then we get a factorable representation of $T[f]$.

Proof:

The proof by induction on the length of the sequence representing f is immediate.

Remark:

The replacement rules are simple and can be carried out mechanically.

Corollary:

Let $f: D \subseteq R^n \rightarrow R^m$ be a piecewise factorable function. Let U_1, U_2, \dots be sets such that $D \subseteq \bigcup U_j$ and $f|_{U_j}$ is a factorable function.

Let $f_{i,1}, f_{i,2}, \dots, f_{i,L_i}$ be basic representation of $f|_{U_i}$ and let $F_{i,1}, \dots, F_{i,L_i}$ be the corresponding "Replacement Sequence". Define F by

$F|_{W_1} = F_{i, L_1}$ where $W_1 = U_1 \times R^{(k-1) \cdot n}$ then F is a piecewise factorable function and $F|_{W_1} = T[f|_{U_1}]$.

4. Taylor Series expansion:

Let \mathcal{F} be a set that consists of the identity function, the arithmetic operations, functions that satisfy first order rational differential equations (like Exp, and log), pairs of functions satisfying second order rational differential equations (like Cos and Sin), and so on.

Choose \mathcal{F} in such a way that all recursion relations between successive derivatives can be expressed as factorable functions.

Let $k \geq 1$ be an integer. We define T_k as follows:

Let $f: W \subseteq R^S \rightarrow R$ be of class C^k . Let $F = T[f]$ be the function satisfying:

a) $F: W \times R^{S \cdot k} \rightarrow R^{k+1}$

b) For every function $X: R \rightarrow R^S$, $X \in C^k$

and every $t_0 \in R$ such that $X(t_0) \in W$

$$F([X(t_0)]_0, [X(t_0)]_1, \dots, [X(t_0)]_k) = ([f(X(t_0))]_0, \dots, [f(X(t_0))]_k).$$

It is not hard to see that the above T_k and \mathcal{F} satisfy the assumptions of Theorem 1.

Note that the set \mathcal{F} contains only analytic functions and therefore the factorable functions are analytic.

5. Gradients:

Let $f_{(j)}$ denote $\frac{\partial f}{\partial x_j}$. Let f be as above. Let $k \geq 1$ be an integer.

Define T_k as follows:

Let $f: W \subseteq \mathbb{R}^S \rightarrow \mathbb{R}$ be of class C^1 and define $F = T[f]$ to be the function satisfying:

- a) $F: W \times \mathbb{R}^{S \cdot k} \rightarrow \mathbb{R}^{k+1}$
- b) For every function $h \in C^1$,
- $$h: \mathbb{R}^k \rightarrow \mathbb{R}^S \quad \text{if} \quad h(X_0) \in W$$

$$\begin{aligned} \text{then } F(h(X_0), h_{(1)}(X_0), \dots, h_{(k)}(X_0)) \\ = (f(h(X_0)), f_{(1)}(h(X_0)), \dots, f_{(k)}(h(X_0))). \end{aligned}$$

Again it is not hard to see that the above f and T_k satisfy the assumptions of Theorem 1.

6. Iterative procedure:

Many functions are computed iteratively. The number of iterations in the computer program must be finite of course. This number however can change according to the values of the arguments. The usual arrangement in iterative procedure is as follows: One prescribes a tolerance ϵ and sometimes an initial guess. The program then proceeds with the iterations until the change in the function value, or the estimated error is $\leq \epsilon$. Once ϵ is fixed, the number of iterations as a function of the arguments is, in most cases, piecewise constant. Since iterative procedure can differ considerably, we cannot say what are the precise conditions that make functions that are computed iteratively, piecewise factorable. However by careful study of a particular problem at hand, in many cases, one can show that the function actually computed is in fact piecewise factorable.

In such a case if one computes derivatives of that function, one actually computes the derivatives of a piecewise factorable function. These derivatives might not be a good approximation to the derivatives we had in mind. However many times one can use the following classical theorem (see [15]).

Theorem: If f_j is a sequence of analytic functions in the complex plane and $f_j \rightarrow f$ uniformly on a closed disc $D(x_0, r)$ then f is analytic in the disc, $f_j^{(k)} \rightarrow f^{(k)}$, uniformly and $\forall z \in D(x_0, r)$

$$\frac{1}{k!} |f_j^{(k)}(z) - f^{(k)}(z)| \leq \frac{1}{r^k} \sup_{w \in D(x_0, r)} |f_j(w) - f(w)|.$$

7. Taylor Series Expansions of Implicit Functions:

Let f be a factorable function $f: \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ and assume that $f(t_0, x_0) = 0$, $t_0 \in \mathbb{R}$, $x_0 \in \mathbb{R}^n$ and that $\Gamma = f_x^{-1}(t_0, x_0)$ exists. Then the system of equations $f(t, x(t)) \equiv 0$ defines implicitly a unique function $x: \mathbb{R} \rightarrow \mathbb{R}^n \ni x(t_0) = x_0$ and $f(t, x(t)) \equiv 0$ in the neighborhood of (t_0, x_0) .

Moreover, from Theorem 20.3 in [6, Ch 5, p. 329-332] it follows that if

$$g_i(t) = f(t, \sum_{j=0}^{i-1} [x(t_0)]_j \cdot (t-t_0)^j)$$

then

$$i) \quad \text{For } m < i \quad [g_i(t_0)]_m = 0$$

$$ii) \quad [x(t_0)]_j = -\Gamma [g_i(t_0)]_i.$$

Clearly $g_i(t)$ is a factorable function and one can compute $[g_i(t_0)]_i$. Therefore one can compute $[x(t_0)]_i$ by the following bootstrap procedure:

Start by the regular Newton method to find x_0 such that $f(t_0, x_0) = 0$ then set $\Gamma = f_x^{-1}(t_0, x_0)$. At this point it is possible to compute $[x(t_0)]_1$. Once one has $[x(t_0)]_1$ one can compute $[x(t_0)]_2$ and so on.

8. Computation of Sparse Gradients

a) Band gradients: Many times the gradient matrix is in band form (for example in the numerical solution of a two-point boundary value problem). In such a case the gradient can be computed with great saving in time and space.

Let $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a factorable function (piecewise factorable), and assume that, for all j , f^j depends only on $\{x^i\}$ where $|i-j| \leq k$. Assume that we want to compute the partial derivatives of f with respect to $x \in \mathbb{R}^n$. Instead of reserving $n+1$ words for each variable and starting with the matrix:

$$n+1 \left\{ \begin{pmatrix} x^1 & x^2 & x^3 & \dots & x^n \\ 1 & 0 & & & 0 \\ 0 & 1 & & & 0 \\ 0 & 0 & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ \vdots & \vdots & & & 1 \end{pmatrix} \right.$$

One needs to reserve only $2k+2$ words for each variable and start with the matrix

$$2k+2 \left\{ \begin{pmatrix} x^1 & x^2 & \dots & x^{2k+1} & x^{2k+2} & x^{2k+3} & \dots & x^n \\ 1 & 0 & & \vdots & 1 & 0 & & \\ 0 & 1 & & \vdots & 0 & 1 & & \\ \vdots & 0 & \ddots & & \vdots & 0 & \ddots & \\ \vdots & \vdots & & \ddots & \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & 1 & 0 & 0 & & \end{pmatrix} \right.$$

b) The general case: Many times the gradient matrix is sparse but does not have a structure that is easy to take advantage of. However it is not necessary to know in advance which of the entries of the matrix are identically zero. One can carry this information with the computation and use the following obvious fact:

If

$$J_t = \{ i \mid \frac{\partial g_t}{\partial x_i} \neq 0 \}$$

and if

$$h(x) = f(g_1(x), \dots, g_s(x))$$

then

$$J = \{ i \mid \frac{\partial h}{\partial x_i} \neq 0 \} \subseteq \bigcup_{t=1}^s J_t.$$

In the implementation, each variable will be a pointer to a vector which will keep a list of nonzero partial derivatives. Each subroutine that replaces a call to a basic library function will compute function values and nonzero partial derivatives only. The subroutine will create a list of the nonzero partial derivatives of the composition.

The implementation of automatic computation of partial derivatives of FORTRAN functions (GRADIENT) described in this report does not use the above method. We plan to implement this method in the near future.

9. The Implementation.

9.1. Introduction.

In earlier sections it was pointed out that most computer functions and subroutines used in numerical computations compute (represent) piecewise factorable functions. It was shown that every sequence representing a piecewise factorable function can be transformed into another sequence which represents the original piecewise factorable function and its total or partial derivatives. The translation process is merely a replacement process and can be carried out mechanically.

In order to implement such a replacement process one needs a processor that will do the following:

- a) Break the subprogram into a sequence of one step terms.
- b) Replace each term by a body of code.
- c) Expand each program variable into a vector. The size of that vector will depend on the order of the operator implemented.
- d) The processor should leave the control structure unaltered, that is: Do loops and IF statements should be left unaltered.

There are three principal ways to implement the replacement process.

- a) Macro expansion.
- b) Replacing each term by a subroutine call.
- c) Using an interpreter.

We chose to use the second method mainly because we could use an existing precompiler: AUGMENT. We also feel that the second method is the most powerful and flexible of the three.

9.2. The AUGMENT Precompiler.

Since we are using AUGMENT as the main tool for the implementation of automatic differentiation, we find it appropriate to give a very short description of the function and use of AUGMENT. However, in order to understand fully how to use it, the user should read [3].

The AUGMENT precompiler was designed to simplify the use of nonstandard data types in FORTRAN. AUGMENT enables one to define new data types and operations. It enables one to write FORTRAN programs using these new data types as though they were standard. AUGMENT input consists of programs written in "extended" FORTRAN, that is; FORTRAN programs using nonstandard data types, operators, and functions. AUGMENT translates the input programs into standard FORTRAN programs with the nonstandard constructs translated into subroutine and function calls. The supporting package (that is, the above subroutines and functions) implement the operations with the nonstandard data types.

In order to implement a new data type with AUGMENT one has to do two things

- 1) write a package of subroutines to implement the operations and functions defined on the new data type,
- 2) write a description deck which describes the new data type.

In the description deck one provides AUGMENT with the following information:

- a) The name(s) of the new data type(s).
- b) The number and type of computer words to reserve for each nonstandard variable.
- c) The set of operations and "standard" functions defined on the new data type.
- d) The names and calling sequences of the subroutines and functions that implement the above operations.
- e) The relations between the new data types and other data types (standard and nonstandard).

For more detailed information see [3].

9.3. GRADIENT and TAYLOR Packages.

This report describes the implementation of two types of differentiation:

- 1) TAYLOR: Automatic Taylor series expansion of FORTRAN functions.
- 2) GRADIENT: Automatic gradient computation of FORTRAN functions.

The automatic differentiation is implemented by providing two new data types: TAYLOR and GRADIENT. The operations defined on the new data types are the arithmetic operations and almost all the standard FORTRAN functions. Each subroutine that implements a nonstandard operation computes (represents) the corresponding factorable function G of Theorem 1.

Suppose one wants to compute the gradient of a function f , which is a function of n variables. One has only to write a FORTRAN function (subroutine) that computes f . In the function or subroutine code, one declares the arguments and other variables (including the function itself) as type GRADIENT. Then one submits this function with the description deck (which is part of the package) to AUGMENT as data. The output from AUGMENT will be the desired subroutine. AUGMENT will translate the function into a FORTRAN subroutine written in ANSI standard FORTRAN, declaring each GRADIENT variable as a REAL vector of dimension $n + 1$ (and each k vector as an $(n + 1) \times k$ REAL array and so on). Each arithmetic operation or function call will be translated to a call to the appropriate subroutine. The translated subroutine together with the subroutines provided by the GRADIENT package will compute the gradient of the function f at any desired point.

Below we give a detailed description of the two packages and their use. Most of the details of the two packages are the same: Anything that is said below applied to both packages, unless the contrary is explicitly stated. We will use the term VARIABLE for either type GRADIENT or TAYLOR and CONSTANT for types REAL, INTEGER or DOUBLE PRECISION. The relations between VARIABLE and type COMPLEX are undefined.

TAYLOR and GRADIENT Variables:

Each GRADIENT variable is a REAL vector of dimension $N + 1$ where N is the number of the independent arguments. The first word holds the variable value and the $(I + 1)$ th word holds the partial derivative of that variable with respect to the I th independent argument.

Each TAYLOR variable is a Real vector of dimension $N + 1$ where N is the highest normalized derivative to be computed. The $(I + 1)$ th place holds the I th normalized derivative, $I = 0, 1, \dots, N$.

Arithmetic Operations:

All arithmetic operations between VARIABLES and all arithmetic operations between VARIABLE and CONSTANT are legal except INTEGER raised to a VARIABLE power. Since the recursion relations that replace arithmetic operations between CONSTANT and VARIABLE are simpler than the general relations, separate routines are provided to implement the arithmetic operations between CONSTANTS and VARIABLES. Conversion of CONSTANT to a vector format is done if there is a statement of the form $V = c$ where V is a VARIABLE and c is a REAL expression, or if there is a reference to a conversion function (see Conversion routines).

Standard Functions:

In Table 1, we list the standard functions that are implemented in the two packages. One can easily add other functions to that list by adding their names to the description deck and writing subroutines to implement them.

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

4 OF 7
ADA
046552

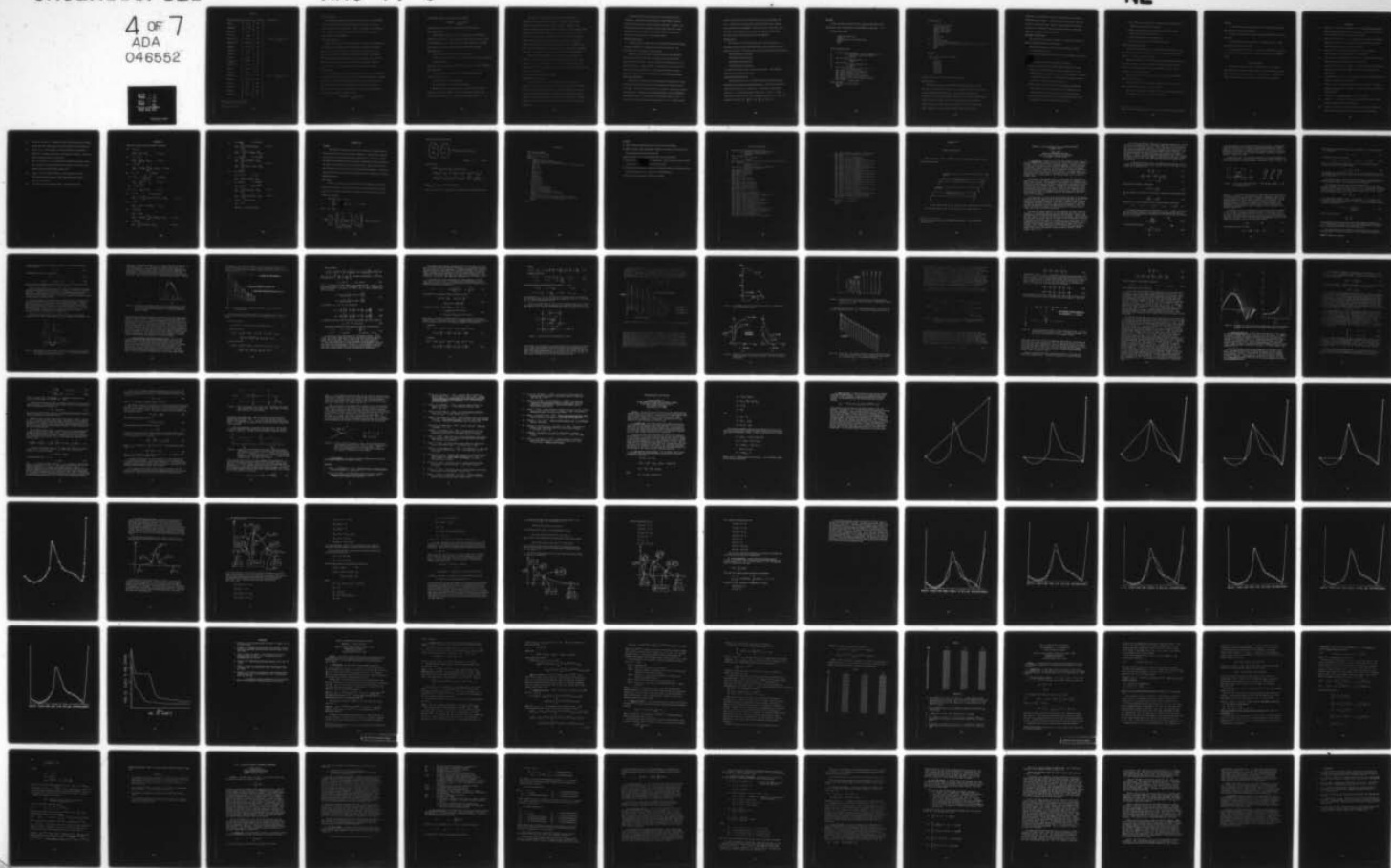


Table 1.

Function reference	Function	Suffix	Comments
ABS(x)	$ X $	ABS	
ACOS(x) [†]	$\cos^{-1}(x)$	ACS	
ALOG(x) [†]	$\ln(x)$	LN	
ALOG10(x)	$\log_{10}(x)$	LOG	
AMAX1(x,y) [†]	$\max(x,y)$	MAX	function of two arguments only
AMIN1(x,y) [†]	$\min(x,y)$	MIN	"
ATAN(x)	$\tan^{-1}(x)$	ATN	
ASIN(x) [†]	$\sin^{-1}(x)$	ASN	
CBRT(x) [†]	$X^{1/3}$	CBR	
COS(x)	$\cos(x)$	COS	
COSH(x) [†]	$\cosh(x)$	CSH	
COTAN(x) [†]	$\cotan(x)$	CTN	
EXP(x)	$\exp(x)$	EXP	
LOG(x) [†]	$\ln(x)$	LN	
MAX(x,y)	$\max(x,y)$	MAX	function of two arguments only
MIN(x,y)	$\min(x,y)$	MIN	"
SIN(x)	$\sin(x)$	SIN	
SINH(x) [†]	$\sinh(x)$	SNH	
SQRT(x)	$X^{\frac{1}{2}}$	SQR	
TAN(x) [†]	$\tan(x)$	TAN	

[†] Not an ANSI standard function.

[‡] See automatic typing.

Automatic Typing:

This package provides the same feature that exists in many FORTRAN compilers, automatic typing of functions; that is, the type of function used is determined by its arguments. Thus, for example, we have defined LOG and ALOG to be names of the function which takes the logarithm of an argument of type VARIABLE.

Conversion Functions:

There are three subroutines that implement conversion from CONSTANT to VARIABLE, one each for types REAL, INTEGER and DOUBLE PRECISION. These routines can be referenced in the original program by the use of the conversion function: CTTYL(.) in TAYLOR and CTGRD(.) in GRADIENT. The function accepts all three standard types as arguments (see automatic typing). Automatic conversion is invoked only for type REAL. That is: the statement $V = \text{const}$ is legal only if the const. is of type REAL.

Norm Function:

It is sometimes convenient to test the distance between two VARIABLES (for example in a test of convergence). Since the relational operators compare only the first words of the VARIABLES they cannot be used for that purpose. The packages provide a function NORM that computes the distance of a VARIABLE from the 0 vector. In TAYLOR package, the function NORM is a function of two arguments, TAYLOR and REAL

$$\text{NORM}(v, t) = \max_{0 \leq i \leq N} |[v]_i| t^i$$

In GRADIENT, NORM is a function of one argument

$$\text{NORM}(V) = \max_{0 \leq i \leq N} (|V_i|) .$$

In both packages the function is implemented as REAL function.

Logical Statements:

The relational operators can be used to compare two VARIABLES, or VARIABLE with type REAL. The comparison is done between the first words of the VARIABLES or between the first word of the VARIABLE and type REAL. The comparison operators are implemented as LOGICAL functions.

Other Subroutines:

The packages provide two additional subroutines:

- 1) Error handling subroutine (see our later discussion of Error handling).
- 2) Copy subroutine.

The copy subroutine implements the statement $A = B$, A and B VARIABLES.

Subroutine Names:

The names of all subroutines in both supporting packages are composed of two parts:

- i) The first three letters (the prefix).
- ii) The last three or two letters (the suffix).

All the routines in each package have a common prefix: TYL in TAYLOR and GRD in GRADIENT. In order to avoid name conflicts, the user should avoid using names starting with the above prefixes.

The suffix of a subroutine's name depends on the function or operation the supporting routine implements. In Table 1, we give the suffixes of the routines implementing the "Standard" functions. The suffix of a routine implementing arithmetic operations is given systematically. The first letter describes the operator. A for +, S for -, M for *, D for /, and E for **. The next two letters describe the operands: first the left operand and then the right one. The letter R stands for REAL, I for INTEGER, D for DOUBLE PRECISION and V for VARIABLE. So MVV will be the suffix of a routine that implements (VARIABLE) * (VARIABLE) and DDV of a routine that implements (DOUBLE PRECISION)/(VARIABLE).

The suffix of the LOGICAL functions implementing the relational operators is composed of the two letters representing the operator and the letter V. So .LT. is implemented by a function with suffix LTV. The suffix of the routine implementing the norm function is NRM, Error routine - ERR and copy routine - CPY.

9.4. Using the Package with AUGMENT.

Writing the Source Code:

In order to get derivatives of a function, say the Taylor series expansion of a function \hat{f} , the user should write an "extended" FORTRAN function or subroutine that computes \hat{f} . All legal FORTRAN constructs can be used. All program variables which depend on the independent variable should be declared as type TAYLOR, including the function itself. Program variables which do not depend on the independent variable can be of any other type.

External functions and subroutines can be used as part of the computation. Functions must be declared as type TAYLOR. External functions and subroutines can be translated separately. However, one has to make sure that the number of computer words reserved for each TAYLOR variable in the external subroutine (function) is the same as the number of words reserved in the calling routine.

The Description Deck:

The next step is to submit the source deck with the description deck as data to AUGMENT. See Appendix C for the deck structure. The description deck is supplied with the package.

However, since the number of words reserved for each VARIABLE changes from problem to problem, this number has to be put into the description deck each time. To make it easier, the description deck was split into two parts: HEAD and BODY. The number of words to reserve is inserted in between the two parts. This number should be an integer constant. Column 1 in the card holding this number must be left blank.

Order of Differentiation:

The routines in both packages were designed to implement any "order" of differentiation without the need to be recompiled every time the "order" is changed. The "order" of differentiation is provided to the package through a common block. In TAYLOR by COMMON/DEGREE/N and in GRADIENT by COMMON/ORDER/N N is the order of the operator, that is: if $N = 1$ the package will compute function values only; if $N = 2$, function values

and first derivatives (or first partial with respect to one variable); and so on. The routines in the package do not check that there is enough space provided for the VARIABLES. However there is a check that $N \geq 1$. The order can be changed at run time but care should be taken not to exceed the number of words provided for each VARIABLE.

Working Space:

Some routines in TAYLOR need work space and, since they are designed to handle any order of differentiation, the work space has to be provided by the user. The work space is provided through four common blocks.

COMMON/WORK1/WORK1(N)

COMMON/WORK2/WORK2(N)

COMMON/WORK3/WORK3(N)

COMMON/WORK4/WORK4(N)

N should be the highest order of differentiation used. The GRADIENT package does not require work space.

Using the Translated Routine:

The translated routine is a FORTRAN subroutine that gets as input the value of its arguments and their derivatives, and gives as output the value of the function and its derivatives. For example, in the Taylor package, if t is the independent variable, then $t, 1, 0, 0, \dots$ is the Taylor series expansion of t . In the Gradient package, if x is the j^{th} independent variable then $\frac{\partial x}{\partial x_j} = 1$ and $\frac{\partial x}{\partial x_i} = 0$ for $i \neq j$.

Example:

In this example we compute the first 9 terms of the Taylor series expansion of $f(t) = \exp(\cos(4t)) + \arctan(\sin^2(t))$ at the point $t = .5$.

a) The source code:

```
TAYLOR FUNCTION FUN(T)
TAYLOR T
FUN=EXP(COS(4.*T))+ATAN(SIN(T)**2)
RETURN
END
```

b) The translated code:

```
SUBROUTINE FUN (T, TYLRLT)
C          ===== PROCESSED BY AUGMENT, VERSION 4L =====
C          ----- TEMPORARY STORAGE LOCATIONS -----
C          TAYLOR
REAL TYLTMP(9,2)
C          ----- LOCAL VARIABLES -----
C          TAYLOR
REAL TYLRES(9)
C          ----- GLOBAL VARIABLES -----
C          TAYLOR
REAL T(9), TYLRLT(9)
C          ===== TRANSLATED PROGRAM =====
CALL TYLHRV (4.,T,TYLTMP(1,1))
CALL TYLCOS (TYLTMP(1,1),TYLTMP(1,1))
CALL TYLEXP (TYLTMP(1,1),TYLTMP(1,1))
CALL TYLSIN (T,TYLTMP(1,2))
CALL TYLEVI (TYLTMP(1,2),2,TYLTMP(1,2))
CALL TYLATN (TYLTMP(1,2),TYLTMP(1,2))
CALL TYLAUV (TYLTMP(1,1),TYLTMP(1,2),TYLRES)
GO TO 30000
C          ----- RETURN CODE -----
30000 CONTINUE
CALL TYLCPY (TYLRES,TYLRLT)
RETURN
END
```


c) Main program:

```
      DIMENSION T(9),FUNC(9)
      COMMON /DEGREE/ N
      COMMON /WORK1/W1(9)
      COMMON /WORK2/W2(9)
      COMMON /WORK3/W3(9)
      COMMON /WORK4/W4(9)
      N=9
      T(1)=.5
      T(2)=1.
10    DO 10 I=3,9
      T(I)=0.
      CALL FUN(T,FUNC)
      PRINT1,T(1),(FUNC(L),L=1,9)
91    FORMAT(1X,'EXAMPLE: TAYLOR SERIES EXPANSION',/,3X,'T=',F6.3//,
      * (3X,E13.5))
      STOP
      END
```

d) Output:

```
EXAMPLE: TAYLOR SERIES EXPANSION
T= .500

.88551400
-.15998101
.69251401
-.77435401
-.34296401
.35406402
-.59899402
.28534402
.10762103
```

In Appendix B we give a more complicated example.

Error Handling:

In general the packages do not check that the arguments are within the domain of definition of the functions. Checking is done only for division by REAL, INTEGER or VARIABLE types. The packages also provide the capability to specify what constitutes division by zero. If the absolute value of an argument to division routine is smaller than or equal to specified value, error occurs. This value is set initially to 0.0 by a DATA statement. However it can be changed in runtime through common block

/ZERO/EPS. The routines in the package also check that the order of differentiation is at least 1. In case an error is discovered, the error routine is called (suffix ERR). The error routine prints error messages and performs a "walk back" (which traces the sequence of subprogram calls back to the main program) and stops.

Non ANSI FORTRAN Parts:

No special care was taken to comply with some of the restrictions imposed by ANSI FORTRAN, for two reasons:

- a) Some of the features in the packages could not have been implemented otherwise.
- b) Some of the restrictions violated seem to us arbitrary and unreasonable.

For they do not exist in most production compilers.

Below we give the list of non ANSI FORTRAN constructions used.

- 1) The set of "standard" functions used is larger than the set in ANSI FORTRAN. Functions which are not in the Standard are flagged in the function table by †. The corresponding routines could be deleted or modified to fit different systems.
- 2) The work space to TAYLOR is provided through common blocks and therefore the common block sizes in the TAYLOR routines would be different from the block sizes in the main program.
- 3) The walk back routine used in the error handling routine is non-standard and has to be changed in other systems.

- 4) The REAL variable EPS appears in common block /ZERO/ and in DATA statement (in the error routine).
- 5) No care was taken to comply with the standard concerning expressions as subscripts to arrays.
- 6) The function ATAN2 is not implemented.
- 7) No care was taken not to mix INTEGER with REAL.

Using the Package; Summary:

Assume one has all the package subroutines in relocatable form, so they can be used as library routines. In order to get the derivatives of a function \hat{f} one has to do the following:

- A) Write a FORTRAN subroutine[†] (function) that computes the function \hat{f} . In the subroutine, declare all FORTRAN variables that depend on the independent variables as type TAYLOR (or GRADIENT). The rest of the variables can be of any other type.
- B) Insert the number of computer words reserved for each VARIABLE into the description deck.
- C) Submit the source deck with the appropriate description deck as data to AUGMENT (See Appendix C and also [3]).
- D) Submit the output from AUGMENT as data to the FORTRAN compiler.
- E) Call the subroutine with the desired arguments.

[†]One can use main programs too but that is a more complicated way of doing it.

Remarks

- 1) Always remember that the initial values of the derivatives of the input variables have to be provided too.
- 2) All the restrictions that apply to the use of AUGMENT apply to the packages.
- 3) AUGMENT does not translate I/O and DATA statements. Their translation has to be done by hand.
- 4) This report is by no means a substitute for AUGMENT user information manual (MRC TSR #1469 [3]). The user should be familiar with that report.

Acknowledgments

The author is indebted to Dr. C. de Boor and Dr. S. V. Parter for many stimulating discussions, constructive criticism and valuable suggestions. The author wishes to thank Dr. F. Crary and Dr. L. Landweber for critical reading of the manuscript and for many valuable suggestions.

References

1. Barton, D. , Willer, I.M. and Zahar, R.V.M. , Taylor Series Method for Ordinary Differential Equations. An Evaluation. Mathematical Software, 1971, Academic Press, Inc. , New York and London.
2. Braun, J. A. and Moore, R. E. , A program for the solution of differential equations using Interval Arithmetic (DIFEQ) for the CDC3600 and 1604. MRC Technical Summary Report #901, June 1968.
3. Crary, F. D. , The AUGMENT Precompiler : I. User Information. MRC Technical Summary Report #1469, December 1974.
4. Crary, F. D. , The AUGMENT Precompiler: II. Technical Documentation, MRC Technical Summary Report #1420, October 1975.
5. Knapp, H. and Wanner, G. LIESE II. A Program for Ordinary Differential Equations using Lie-Series, MRC Technical Summary Report #1008, March 1970.
6. Krasnosel'skii, M. A. et. al. , Approximate Solution of Operator Equations, Wolters-Noordhoff, Groningen, 1972.
7. Moore, R. E. , The Automatic Analysis and Control of Error in Error in Digital Computation, Vol. 1, Editor L. B. Rall, John Wiley and Sons, Inc. New York 1965.
8. Moore, R. E. , Interval Analysis, Prentice Hall 1966.
9. Pugh, R. E. , A language for nonlinear programming problems, Mathematical Programming 2 (2) 176-206, 1972.
10. Reiter, A. Automatic generation of Taylor coefficients (TAYLOR) for the CDC 1604, MRC Technical Summary Report #830, November 1967.

11. Reiter, A, and Gray J. , Compiler of Differentiable Expressions (CODEX) for the CDC 3600, MRC Technical Summary Report #791, December 1967.
12. Wertz, H. J. , SUPER-CODEX: Analytic Differentiation of FORTRAN Statements. Aerospace Corporation, Los Angeles, California. Aerospace report No TOR-0172(9320)-12, June 1972.
13. Kuba, D. and Rall, L. B. , A UNIVAC 1108 program for obtaining rigorous error estimates for approximate solution of systems of equations, MRC Technical Summary Report #1168, January 1972.
14. Gray, J. and L. B. Rall, NEWTON: A general purpose program for solving nonlinear systems. MRC Technical Summary Report #790, September 1967.
15. W. Rudin, Real and Complex Analysis. McGraw-Hill, 1966.

Appendix A

Recurrence Relations for Taylor Series Expansion:

a) $Z = X \pm Y$

$$[Z]_J = [X]_J \pm [Y]_J \quad J = 0, 1, \dots$$

b) $Z = X \cdot Y$

$$[Z]_J = \sum_{k=0}^J [X]_k \cdot [Y]_{J-k} \quad J = 0, 1, \dots$$

c) $Z = X/Y$

$$[Z]_J = 1/Y \{ [X]_J - \sum_{k=0}^{J-1} [Z]_k \cdot [Y]_{J-k} \} \quad J = 0, 1, \dots$$

d) $Z = X^I$ I integer

1) $I > 0, \quad I = \sum_{S=0}^n L_S 2^S \quad L_S \in \{0, 1\}$

$$[Z]_J = \left[\prod_{S=0}^n X^{L_S \cdot 2^S} \right]_J \quad J = 0, 1, \dots$$

2) $I = 0; \quad Z \equiv 1$

3) $I < 0 \quad [Z]_J = [1/X^{-1}]_J \quad J = 0, 1, \dots$

e) $Z = X^a$ a real constant

$$[Z]_J = 1/X \cdot \sum_{k=0}^{J-1} ((a(J-k)-k)/J) \cdot [Z]_k \cdot [X]_{J-k} \quad J = 1, 2, \dots$$

f) $Z = X^Y$

$$[Z]_J = [\text{EXP}(Y \cdot \text{LOG}(X))]_J \quad j = 0, 1, \dots$$

g) $Z = \text{LOG}(X)$

$$[Z]_1 = [X]_1 / X$$

$$[Z]_J = 1/X \{ [X]_J - \sum_{k=0}^{J-1} ((J-k)/J) [X]_{J-k} \cdot [Z]_k \} \quad J = 2, 3, \dots$$

h) $Z = \text{EXP}(X)$

$$[Z]_J = \sum_{k=0}^{J-1} ((J-k)/J) [Z]_k \cdot [X]_{J-k} \quad J = 1, 2, \dots$$

- i) $Z = \sin(X), \quad W = \cos(X)$
 $[Z]_J = \sum_{k=0}^{J-1} ((J-k)/J) [W]_k \cdot [X]_{J-k} \quad J = 1, 2, \dots$
 $[W]_J = - \sum_{k=0}^{J-1} ((J-k)/J) [Z]_k \cdot [X]_{J-k}$
- j) $Z = \sinh(X), \quad W = \cosh(X)$
 $[Z]_J = \sum_{k=0}^{J-1} ((J-k)/J) [W]_k \cdot [X]_{J-k} \quad J = 1, 2, \dots$
 $[W]_J = \sum_{k=0}^{J-1} ((J-k)/J) \cdot [Z]_k \cdot [X]_{J-k}$
- k) $Z = \operatorname{ATAN}(X)$
 Let $V = X^2 + 1 \quad W = 1/V$
 $[Z]_J = \sum_{k=0}^{J-1} ((J-k)/J) [W]_k \cdot [X]_{J-k}$
- l) $Z = \operatorname{ASIN}(X), \quad Y = \operatorname{ACOS}(X)$
 Let $V = 1 - X^2 \quad W = 1/\sqrt{V}$
 $[Z]_J = \sum_{k=0}^{J-1} ((J-k)/J) [W]_k \cdot [X]_{J-k} \quad J = 1, 2, \dots$
 $[Y]_J = - \sum_{k=0}^{J-1} ((J-k)/J) [W]_k \cdot [X]_{J-k} \quad J = 1, 2, \dots$
- m) $\operatorname{TAN}(X) = \sin(X)/\cos(X)$
 $\sqrt{x} = x^{\frac{1}{2}}$
 $\operatorname{TANH}(X) = \sinh(X)/\cosh(X)$

Appendix B

Example

The following example was taken from a homework assignment given in an optimization class at the University of Wisconsin. This example is simple but quite typical of problems that arise in applications. We have to compute the gradient of a function which can be easily described by a computer program, but whose explicit expression is quite hard to obtain. When one tries to compute the gradient numerically, one runs into convergence problems. In this example we show how easy it is to get the gradient of such a function by using the GRADIENT package.

The Problem

We have a missile on the north pole of a ball of radius 1 and we want to fly to the south pole. (The units are chosen in such a way that all the constants are 1). Because the problem is symmetric we only have to solve a two dimensional problem.

The motion equations are:

$$\frac{d^2 \mathbf{r}}{dt^2} = -\frac{\mathbf{r}}{|\mathbf{r}|^3} + \mathbf{u} \quad \mathbf{r} = (x, y)$$

Let $t = \alpha \tau$ then

$$\frac{d}{d\tau} \begin{pmatrix} x \\ y \\ \xi \\ \eta \end{pmatrix} = \alpha \left[\begin{pmatrix} \xi \\ \eta \\ \frac{-x}{(x^2 + y^2)^{3/2}} \\ \frac{-y}{(x^2 + y^2)^{3/2}} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ u_1 \\ u_2 \end{pmatrix} \right] = \alpha f(x, y, \xi, \eta, u_1, u_2).$$

Discretize by the Euler method

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \xi_{k+1} \\ \eta_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ \xi_k \\ \eta_k \end{pmatrix} + h \alpha f(x_k, y_k, \xi_k, \eta_k, u_{1,k}, u_{2,k})$$

$$k = 0, 1, \dots, g \quad h = 0, 1.$$

Finally, solve:

$$\begin{aligned} & \min h(\alpha, u_{1,0}, u_{2,0}, u_{1,8}, u_{2,8}, u_{1,9}, u_{2,9}) . \\ & = \{ 20[(x_{10})^2 + (y_{10} + 1)^2 + (\xi_{10})^2 + (\eta_{10})^2 + (u_{1,0}^2 + (u_{2,0})^2 \\ & \quad + (u_{1,8})^2 + (u_{2,8})^2 + (u_{1,9})^2 + (u_{2,9})^2 + \frac{\alpha^2}{10} + \frac{20}{x_5^2 + y_5^2} \} . \end{aligned}$$

$$\text{All } u_{1,j} \equiv 0 \quad u_{2,j} \equiv 0 \quad \text{for } 1 \leq j < 8.$$

Use the variable metric algorithm to solve the above minimization problem.

1. INPUT DECK

```

@XQT MRC*LIB,AUGMENT
@ADD GK*DIFF,DESC-GRD/HEAD
      8
@ADD GK*DIFF,DESC-GRD/BODY
*BEGIN
:FOR,IS TEST1
      IMPLICIT GRADIENT (A-H,O-Z)
      GRADIENT FUNCTION FLTFN(ALFA,U0,US,U9)
      DIMENSION U0(2), US(2),U9(2),U(2,10), X(11),Y(11),UY(11),UX(11)
      X(1)=0.0
      Y(1)=1.0
      UX(1)=0.0
      UY(1)=0.0
      H=.1
      DO 10 I=1,10
      U(1,I)=0.0
      U(2,I)=0.0
10    CONTINUE
      U(1,1)=U0(1)
      U(2,1)=U0(2)
      U(1,9)=US(1)
      U(2,9)=US(2)
      U(1,10)=U9(1)
      U(2,10)=U9(2)
      DO 20 J=1,10
      RS=(X(I)**2+Y(I)**2)**1.5
      X(I+1)=X(I)+H*ALFA*UX(I)
      Y(I+1)=Y(I)+H*ALFA*UY(I)
      UX(I+1)=UX(I)+H*ALFA*(U(1,I)-X(I)/RS)
      UY(I+1)=UY(I)+H*ALFA*(U(2,I)-Y(I)/RS)
20    CONTINUE
      FLTFN=20.*(X(11)**2+(Y(11)+1)**2+UX(11)**2+UY(11)**2)
      $  + U0(1)**2+U0(2)**2+US(1)**2+US(2)**2+U9(1)**2+U9(2)**2
      $  +(ALFA**2)/10.0 +20.0/(X(6)**2+Y(6)**2)
      RETURN
      END
*END

```

Remarks.

- 1) @XQT MRC*LIB.AUGMENT starts the execution of AUGMENT.
- 2) @ADD GK*DIFF.DESC-GRD/HEAD, adds the card image of the HEAD part of the description deck into the run stream.

Similarly @ADD GK*DIFF.DESC-GRD/BODY Adds the BODY part.

- 3) The function is passed to a subroutine. The function and gradient values are stored in the argument of the subroutine.
- 4) The translated subroutine can be used to compute function and gradient values or function values alone. (See Order of Differentiation)
- 5) The rest of the computation details are omitted.

OUTPUT FROM AUGMENT

```

SUBROUTINE FLTFN (ALFA,U0,U8,U9, GRDRLT)
C      ===== PROCESSED BY AUGMENT, VERSION 4H =====
C      ----- TEMPORARY STORAGE LOCATIONS -----
C      GRADIENT
REAL GRDTMP(8,3)
C      ----- LOCAL VARIABLES -----
INTEGER I
C      GRADIENT
REAL H(8), RS(8), U(8,2,10), VX(8,11), VY(8,11), X(8,11), Y(8,11),
*      GRDRES(8)
C      ----- GLOBAL VARIABLES -----
C      GRADIENT
REAL ALFA(8), U0(8,2), U8(8,2), U9(8,2), GRDRLT(8)
C      ===== TRANSLATED PROGRAM =====
CALL GRDCFR (0.0,X(1,1))
CALL GRDCFR (1.0,Y(1,1))
CALL GRDCFR (0.0,VX(1,1))
CALL GRDCFR (0.0,VY(1,1))
CALL GRDCFR (.1,H)
DO 10 I=1,10
CALL GRDCFR (0.0,U(1,1,I))
CALL GRDCFR (0.0,U(1,2,I))
10 CONTINUE
CALL GRDCPY (U0(1,1),U(1,1,1))
CALL GRDCPY (U0(1,2),U(1,2,1))
CALL GRDCPY (U8(1,1),U(1,1,9))
CALL GRDCPY (U8(1,2),U(1,2,9))
CALL GRDCPY (U9(1,1),U(1,1,10))
CALL GRDCPY (U9(1,2),U(1,2,10))
DO 20 I=1,10
CALL GRDEVI (X(1,I),2,GRDTMP(1,1))
CALL GRDEVI (Y(1,I),2,GRDTMP(1,2))
CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
CALL GRDEVV (GRDTMP(1,2),1.5,RS)
CALL GRDMVV (H,ALFA,GRDTMP(1,1))
CALL GRDMVV (GRDTMP(1,1),VX(1,I),GRDTMP(1,1))
CALL GRDAVV (X(1,I),GRDTMP(1,1),X(1,I+1))
CALL GRDMVV (H,ALFA,GRDTMP(1,1))
CALL GRDMVV (GRDTMP(1,1),VY(1,I),GRDTMP(1,1))
CALL GRDAVV (Y(1,I),GRDTMP(1,1),Y(1,I+1))
CALL GRDMVV (H,ALFA,GRDTMP(1,1))
CALL GRDDVV (X(1,I),RS,GRDTMP(1,2))
CALL GRDSVV (U(1,1,I),GRDTMP(1,2),GRDTMP(1,2))
CALL GRDMVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
CALL GRDAVV (VX(1,I),GRDTMP(1,2),VX(1,I+1))
CALL GRDMVV (H,ALFA,GRDTMP(1,1))
CALL GRDDVV (Y(1,I),RS,GRDTMP(1,2))
CALL GRDSVV (U(1,2,I),GRDTMP(1,2),GRDTMP(1,2))

```

```

20      CALL GRDMVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDAVV (VY(1,I),GRDTMP(1,2),VY(1,I+1))
      CONTINUE
      CALL GRDEVI (X(1,11),2,GRDTMP(1,1))
      CALL GRDAVI (Y(1,11),1,GRDTMP(1,2))
      CALL GRDEVI (GRDTMP(1,2),2,GRDTMP(1,2))
      CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDEVI (VX(1,11),2,GRDTMP(1,1))
      CALL GRDAVV (GRDTMP(1,2),GRDTMP(1,1),GRDTMP(1,1))
      CALL GRDEVI (VY(1,11),2,GRDTMP(1,2))
      CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDMRV (20.,GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDEVI (UO(1,1),2,GRDTMP(1,1))
      CALL GRDAVV (GRDTMP(1,2),GRDTMP(1,1),GRDTMP(1,1))
      CALL GRDEVI (UO(1,2),2,GRDTMP(1,2))
      CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDEVI (US(1,1),2,GRDTMP(1,1))
      CALL GRDAVV (GRDTMP(1,2),GRDTMP(1,1),GRDTMP(1,1))
      CALL GRDEVI (US(1,2),2,GRDTMP(1,2))
      CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDEVI (U9(1,1),2,GRDTMP(1,1))
      CALL GRDAVV (GRDTMP(1,2),GRDTMP(1,1),GRDTMP(1,1))
      CALL GRDEVI (U9(1,2),2,GRDTMP(1,2))
      CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,2),GRDTMP(1,2))
      CALL GRDEVI (ALFA,2,GRDTMP(1,1))
      CALL GRDVR (GRDTMP(1,1),10.0,GRDTMP(1,1))
      CALL GRDAVV (GRDTMP(1,2),GRDTMP(1,1),GRDTMP(1,1))
      CALL GRDEVI (X(1,6),2,GRDTMP(1,2))
      CALL GRDEVI (Y(1,6),2,GRDTMP(1,3))
      CALL GRDAVV (GRDTMP(1,2),GRDTMP(1,3),GRDTMP(1,3))
      CALL GRDRV (20.0,GRDTMP(1,3),GRDTMP(1,3))
      CALL GRDAVV (GRDTMP(1,1),GRDTMP(1,3),GRDRES)
      GO TO 30000

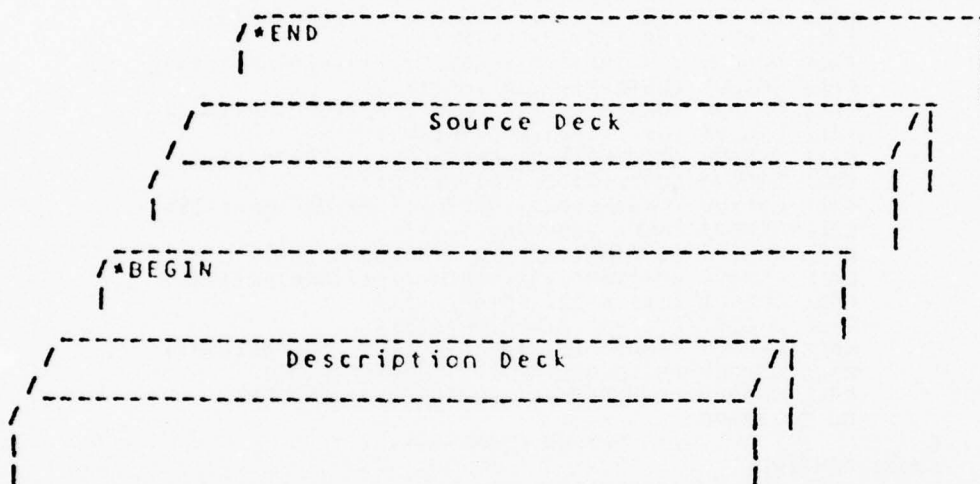
C      ----- RETURN CODE -----
30000 CONTINUE
      CALL GRDCPY (GRDRES,GRDRLT)
      RETURN
      END

```

Appendix C[†]

Deck Arrangement

The data deck read by AUGMENT has the structure shown in the following diagram:



At the conclusion of processing, the translated program decks are in the output file in 80 column card image format.

[†]This page is taken out of: The AUGMENT Precompiler 1. User Information. Fred Crary [3].

PROGRESS IN THE CALCULATION OF TWO- AND THREE-DIMENSIONAL BOUNDARY LAYERS

Tuncer Cebeci
Mechanical Engineering Department
California State University at Long Beach
Long Beach, California 90846

1. INTRODUCTION. Boundary layers, sometimes called thin shear layers [1], occur in flow past bodies (external flows), or flow through ducts or channels (internal flows), at all but very low Reynolds numbers. The major portion of the flow can be considered inviscid since the magnitude of the viscous force is negligible. However, owing to the need to satisfy the no-slip boundary condition imposed by viscous flow, the fluid must have a rapid variation from the inviscid velocity to the zero value at the wall. The region where this variation occurs is restricted to a thin layer close to the wall, of an extent proportional to an inverse power of the Reynolds number, and is called the boundary layer.

The presence of the boundary layer on a body allows the computation of quantities of great importance in the design of vehicles, for example, viscous drag and heat transfer. The phenomenon of flow separation is intimately connected with the processes occurring within the boundary layer where the already retarded flow first reverses direction. Although all of the desired results may be obtained from a direct solution of the Navier-Stokes equations, these solutions are complicated and very costly. They are also unnecessary since at high Reynolds numbers the equations can be simplified to give the boundary-layer equations, which are extremely versatile, and encompass a very wide range of flow situations.

Since the advent of the use of finite-difference methods for the solution of the boundary-layer equations, great advances have been made in both solution techniques, and basic understanding of the flows. Without the use of the computer, our knowledge, especially in the area of turbulent flows, would be considerably less than it is today. Most of the previous work in the field has been concentrated on either two-dimensional or axisymmetric flows. Only recently has there been interest in three-dimensional or unsteady boundary layers, and to date, not much has been done.

The numerical methods used for all these solutions fall into three basic categories. Of historical interest are the difference-differential, or line, methods which utilize shooting techniques for their solution. These are not used much anymore. Of the two true finite-difference methods, both are implicit, with the major distinction being in the number of node points used to form the differences on grid lines, and both can difference across node points. The Box scheme, devised by H. B. Keller [2], uses only two point differences, thus differencing only between grid points, where the difference formulas are also centered. Both of these schemes are in widespread use today, but I will concentrate only on the Box scheme which H. B. Keller and I have used with success over the past ten years on a great variety of problems, see for example, refs. 3 to 6.

As I previously mentioned, standard two-dimensional boundary-layer problems have been well documented, so I will report on two areas of current interest which still leave many open questions. First, the calculation of separating and reverse two-dimensional flows using the boundary-layer equations will be discussed and comparisons with some Navier-Stokes solutions presented. Next the three-dimensional flow past wings will be described and the related mathematical problem of unsteady two-dimensional boundary-layer flow will be presented. Finally, the various models of turbulence will be briefly reviewed and an example of the use of the model we prefer at Douglas Aircraft Company, Long Beach, will be given, along with some flows where more complex models are called for.

2. SEPARATING AND REVERSE FLOWS. The boundary-layer equations for two-dimensional incompressible laminar and turbulent flows with the eddy viscosity concept can be written as [1]

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (1)$$

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{1}{\rho} \frac{dp}{dx} + \frac{\partial}{\partial y} \left(\tilde{b} \frac{\partial u}{\partial y} \right) \quad (2)$$

$$\tilde{b} = (\nu + \epsilon_m) \frac{\partial u}{\partial y} \quad (3)$$

Here the eddy viscosity is defined by

$$-\rho \overline{u'v'} = \rho \epsilon_m \frac{\partial u}{\partial y} \quad (4a)$$

and the pressure is related to the velocity at the edge of the boundary layer by

$$-\frac{1}{\rho} \frac{dp}{dx} = u_e \frac{du_e}{dx} \quad (4b)$$

Equations (1) to (3) are subject to the following boundary conditions:

$$y = 0, \quad u = 0, \quad v = v_w(x); \quad y = y_e, \quad u = u_e(x) \quad (5)$$

The boundary-layer problem as formulated by (1) to (5), where the external velocity (pressure gradient) is prescribed, is termed the standard problem. Once this is solved, quantities of importance can be obtained from the solution such as skin friction

$$c_f = \frac{\mu \left(\frac{\partial u}{\partial y} \right)_w}{\frac{1}{2} \rho u_e^2} \quad (6a)$$

or displacement thickness,

$$\delta^* = \int_0^{y_e} \left(1 - \frac{u}{u_e} \right) dy \quad (6b)$$

It has been found that for the standard problem, as separation is approached, a singularity develops in the solution and the procedure cannot pass through separation into the region of separated (reverse) flow [7,8]. Thus, prior to dealing with the problem of integrating into a negative velocity, a means must be found to enable one to pass through the separation singularity in order to reach this reverse flow region.

The dominant terms in (2) are underlined above, and thus the equation is classified as parabolic. The numerical solution of this equation together with (1) and (5) requires a marching procedure. At stations A and E of Figure 1, the

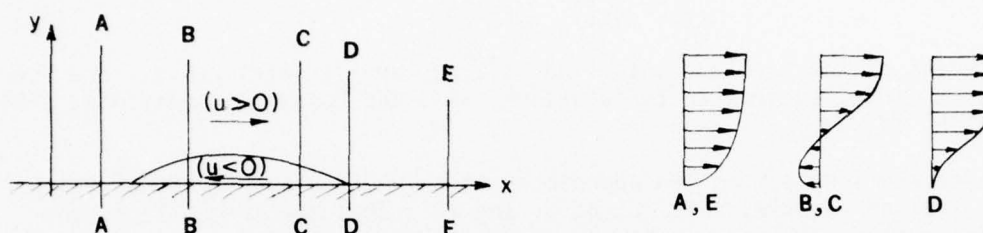


Figure 1. Flow with a "separation bubble." Flow separates between A and B and reattaches at D.

velocity, u , is uniformly positive across the boundary layer, and no difficulties are encountered in the solution procedure. However, at Stations B and C, the velocity has a reverse flow in a portion of the boundary layer, and a marching procedure from B to C cannot be used since the information is being propagated from C to B against the proposed direction of the march. The numerical method will blow up, and indeed the differential equation has exponentially growing solutions if marched into a region of negative flow. Consequently, something must be done if a solution is to be obtained in the region of reverse flow.

This difficulty can be avoided by using the so-called inverse procedures [5-10]. Instead of imposing the pressure gradient as known (which implies u known), some other quantity, such as C_f or δ^* , is assumed given and the correct pressure gradient, consistent with this quantity and the governing equations, is deduced. As a first step, rewrite the equations in terms of the stream function which satisfies the continuity equation

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \quad (7)$$

The momentum equation (2) becomes

$$(b\psi'')' - \frac{1}{\rho} \frac{dp}{dx} = \psi' \frac{\partial \psi'}{\partial x} - \psi'' \frac{\partial \psi}{\partial x} \quad (8)$$

Here the primes denote differentiation with respect to y and the parameter b is defined by

$$b = v + \epsilon_m \quad (9)$$

The boundary conditions (5) become

$$y = 0, \quad \psi = \psi' = 0; \quad y = y_e \quad \psi'_e = u_e(x) \quad (10a)$$

If δ^* is also given, then from (6b), at $y = y_e$ we can impose another boundary condition on the system, namely,

$$\psi_e = u_e[y_e - \delta^*(x)] \quad (10b)$$

The statement of the problem is to find $\psi(x,y)$ and its derivatives, and the pressure gradient, subject to the equation, (8), and boundary conditions, (10) for a given $\delta^*(x)$.

Keller and I have used two approaches to solve the inverse boundary-layer problem, see for example, refs. 5, 6, 9, and 10. They are termed "the nonlinear eigenvalue method", and "the Mechul* function method."

The procedure of the nonlinear eigenvalue method [5,10] is as follows. For each x -location, assume a value of p_x , call it β . Now solve the standard problem using the value of u_e given by this β , by the Box method. From this solution compute a value of the displacement thickness, call it δ_C^* . Compare this with the given value of δ^* , i.e.,

$$\phi(\beta) \equiv \delta_C^* - \delta^* \quad (11)$$

If β were chosen correctly, then ϕ would be zero. If it is not zero to within some small tolerance, a new, better guess for β is required. This is found by finding the variation of ϕ with β , and using a Newton procedure to predict a new value for β

$$\beta^{v+1} = \beta^v - \frac{\phi(\beta^v)}{\frac{\partial}{\partial \beta} [\phi(\beta^v)]} \quad (12)$$

Using (11), we can write

$$\frac{\partial \phi}{\partial \beta} = \frac{\partial \delta_C^*}{\partial \beta} \quad (13)$$

This requires the solution of a set of variational equations obtained by differentiating the differenced equations of the standard problem with respect to β . The iteration procedure is continued until convergence is obtained, usually requiring no more than two or three iterations.

The second approach using the Mechul-function method [6,9], uses the momentum equation as previously given, eq. (8), but treats the pressure as an

*Mechul: Turkish for unknown.

unknown (hence the name of the method), and adds, as an additional equation to close the set,

$$p' = 0 \quad (14)$$

The boundary conditions are specified as

$$y = 0: \quad \psi = \psi' = 0 \quad (15a)$$

$$y = y_e: \quad \psi = u_e(y_e - \delta^*), \quad \psi' = u_e \quad (15b)$$

The Box method using Newton's iteration is used to solve this set which again converges in two or three iterations at each x step.

Both of these methods have been used to solve various laminar and turbulent boundary-layer problems where no separation is present. The results of each are comparable. However, when a separation zone is encountered, and an attempt is made to solve the equations, as below, only the Mechul function method is successful. Thus, in what follows, it is only this method which has been modified to pass through the separation point.

The major problem to be overcome in the separation region is the advance of the solution into the oncoming reverse flow in the boundary layer. When this velocity is small, as it usually is in limited separation bubbles, an approximation made by Reyhner and Flugge-Lotz [11], called FLARE [12], can be used. This consists of setting the value of u equal to zero wherever $u < 0$ within the boundary layer. Thus, the entire difficulty of marching into an oncoming flow is eliminated. Incorporating this approximation into the inverse procedures, allows the computation of separated flows.

The results of such calculations are given in Figure 2. Here a case computed using the Navier-Stokes equations [13] was recomputed using Mechul with

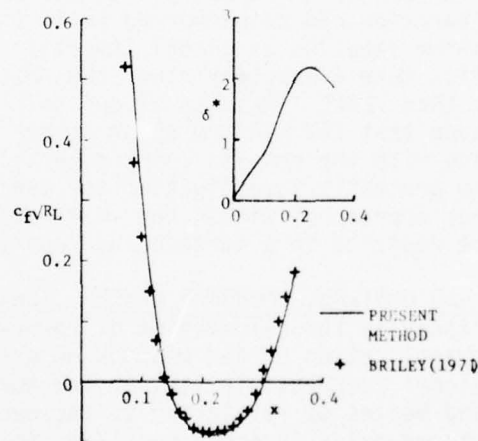


Figure 2. Calculated local skin-friction coefficient distribution for separating and reattaching flow computed by Briley. The present method denotes the solutions obtained by Mechul [9].

FLARE [9]. The prescribed value of δ^* is shown, along with a comparison of the computed values of cf . The agreement is obviously very good. As a further demonstration of the ability of the method, another separated flow previously computed by a boundary-layer technique [14] was recomputed using Mechul-FLARE. The original calculations were also made with the FLARE approximation, but using a modified set of equations for an inverse boundary layer [14]. These are shown in Figure 3 labelled forward marching. A further

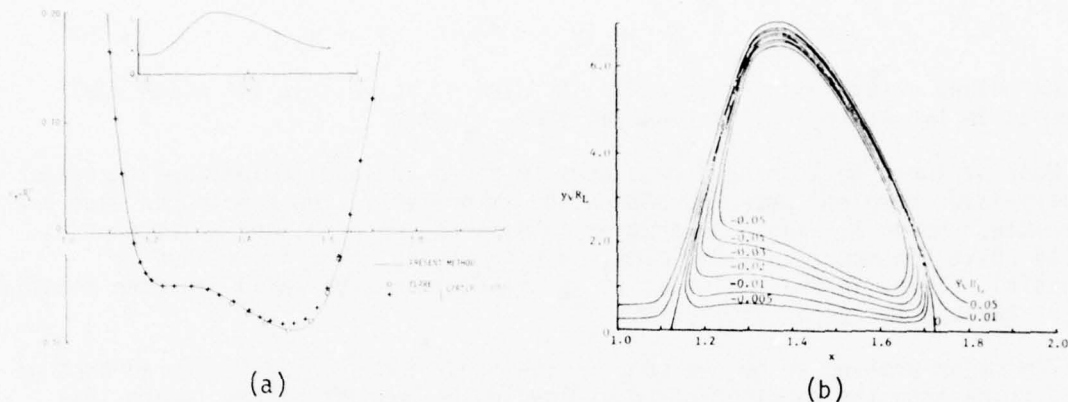


Figure 3. Comparison of calculated results for the flow computed by Carter. The present method denotes the solutions obtained by Mechul [9]. (a) Local skin friction distribution. (b) Streamline pattern in separation bubble.

improvement over FLARE can be made once a complete pass through the separated region is finished. With the computed velocities obtained from FLARE, the convection terms involving u can be differenced with the prevailing wind of the previous pass, and an iteration can be performed until the results do not change [9,14]. These are shown labelled as global iteration in Figure 3, and should represent the true flow more accurately since the physical situation is modelled somewhat better than FLARE. Results of our solution procedure are also given, and it can be seen that the minimum shear agrees quite well with the previous global iteration with the results being generally between the two previous calculations. I am presently investigating the use of this global iteration procedure and other approaches within the framework of the Mechul method. The results will be reported in a forthcoming report.

3. THREE-DIMENSIONAL AND UNSTEADY BOUNDARY LAYERS. One of the most important problems we face today is the calculation of three-dimensional boundary layers. The major application of the methods we use is to the calculation of three-dimensional compressible laminar and turbulent boundary layers on arbitrary wings and bodies of revolution at incidence. One obstacle that had to be overcome prior to other considerations, for the calculation of boundary layers on wings, was the choice of a coordinate system in which to perform the boundary layers calculations. The one we finally chose as being the most appropriate for wings is the body-oriented

nonorthogonal system shown in Figure 4. Using this system the entire wing can easily be covered with grid points which follow the exact plan form of the wing [15]. Methods that use orthogonal systems are not very suitable for wings as discussed in ref. 15.

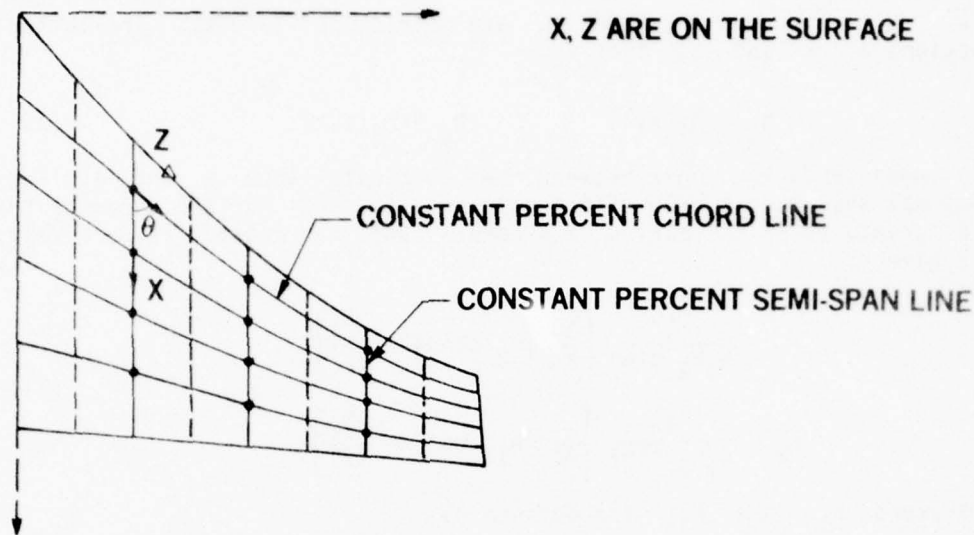


Figure 4. The nonorthogonal system used by Cebeci, Kaups and Ramsey for wing calculations.

The governing boundary-layer equations for three-dimensional compressible laminar and turbulent boundary layers for a nonorthogonal system can be written as [15]:

Continuity equation

$$\frac{\partial}{\partial x} (\rho u h_2 \sin \theta) + \frac{\partial}{\partial z} (\rho w h_1 \sin \theta) + \frac{\partial}{\partial y} (\rho \bar{v} h_1 h_2 \sin \theta) = 0 \quad (16)$$

x-Momentum equation

$$\begin{aligned} \rho \frac{u}{h_1} \frac{\partial u}{\partial x} + \rho \frac{w}{h_2} \frac{\partial u}{\partial z} + \rho \bar{v} \frac{\partial u}{\partial y} - \rho \cot \theta K_1 u^2 + \rho \csc \theta K_2 w^2 + \rho K_{12} u w \\ = - \frac{\csc^2 \theta}{h_1} \frac{\partial p}{\partial x} + \frac{\cot \theta \csc \theta}{h_2} \frac{\partial p}{\partial z} + \frac{\partial}{\partial y} \left(\mu \frac{\partial u}{\partial y} - \rho \overline{u'v'} \right) \end{aligned} \quad (17)$$

z-Momentum equation

$$\begin{aligned} \rho \frac{u}{h_1} \frac{\partial w}{\partial x} + \rho \frac{w}{h_2} \frac{\partial w}{\partial z} + \rho \bar{v} \frac{\partial w}{\partial y} - \rho \cot \theta K_2 w^2 + \rho \csc \theta K_1 u^2 + \rho K_{21} u w \\ = \frac{\cot \theta \csc \theta}{h_1} \frac{\partial p}{\partial x} - \frac{\csc^2 \theta}{h_2} \frac{\partial p}{\partial z} + \frac{\partial}{\partial y} \left(\mu \frac{\partial w}{\partial y} - \rho \overline{w'v'} \right) \end{aligned} \quad (18)$$

Energy equation

$$\rho \frac{u}{h_1} \frac{\partial H}{\partial x} + \rho \frac{w}{h_2} \frac{\partial H}{\partial z} + \overline{\rho v} \frac{\partial H}{\partial y} = \frac{\partial}{\partial y} \left[\frac{\mu}{Pr} \frac{\partial H}{\partial y} + \mu \left(1 - \frac{1}{Pr} \right) \frac{\partial}{\partial y} \left(\frac{u_t^2}{2} \right) - \rho \overline{v'H'} \right] \quad (19)$$

Here $\overline{\rho v} = \rho v + \overline{\rho'v'}$ and h_1 and h_2 are metric coefficients. The latter are functions of x and z , that is,

$$h_1 = h_1(x, z) \quad h_2 = h_2(x, z) \quad (20)$$

Also, θ represents the angle between the coordinate lines x and z . For an orthogonal system $\theta = \pi/2$. The parameters K_1 and K_2 are known as the geodesic curvatures of the curves $z = \text{const.}$ and $x = \text{const.}$, respectively. They are given by

$$K_1 = \frac{1}{h_1 h_2 \sin \theta} \left[\frac{\partial}{\partial x} (h_2 \cos \theta) - \frac{\partial h_1}{\partial z} \right] \quad (21)$$

$$K_2 = \frac{1}{h_1 h_2 \sin \theta} \left[\frac{\partial}{\partial z} (h_1 \cos \theta) - \frac{\partial h_2}{\partial x} \right]$$

The parameters K_{12} and K_{21} are defined by

$$K_{12} = \frac{1}{\sin \theta} \left[- \left(K_1 + \frac{1}{h_1} \frac{\partial \theta}{\partial x} \right) + \cos \theta \left(K_2 + \frac{1}{h_2} \frac{\partial \theta}{\partial z} \right) \right] \quad (22a)$$

$$K_{21} = \frac{1}{\sin \theta} \left[- \left(K_2 + \frac{1}{h_2} \frac{\partial \theta}{\partial z} \right) + \cos \theta \left(K_1 + \frac{1}{h_1} \frac{\partial \theta}{\partial x} \right) \right] \quad (22b)$$

u_t represents the total velocity within the boundary layer and is given by

$$u_t = (u^2 + w^2 + 2uw \cos \theta)^{1/2} \quad (22c)$$

The boundary conditions for the system of equations (16) to (19) are:

$$\begin{aligned} y = 0: \quad u = v = w = 0, \quad \left(\frac{\partial H}{\partial y} \right) &= 0 \\ y = \delta: \quad u = u_e(x, z), \quad w = w_e(x, z) \quad H = H_e \end{aligned} \quad (23)$$

In addition to closure assumptions for the Reynolds stresses $-\rho \overline{u'v'}$, $-\rho \overline{v'w'}$ and $-\rho \overline{v'H'}$, equations (17) to (19) also require initial conditions on two intersecting planes. One of these is the y - z plane along the initial x -station where data is first prescribed, while the other is an x - y plane away from which the subcharacteristics of the hyperbolic crossflow allow one to march the equations. In fact, it is the presence of the cross-flow terms in the equations which distinguishes the calculation of three-dimensional boundary layers from their two-dimensional counterparts.

The boundary-layer equations as given by (16) to (19) can be solved when they are expressed either in physical coordinates or in transformed coordinates. Each coordinate has its own advantages. In three-dimensional flows, where the computer storage and time becomes quite important, the choice of using transformed coordinates becomes necessary as well as convenient because the transformed coordinates allow large steps to be taken in the streamwise and spanwise directions. In addition, they remove the singularity at $x = 0$ and $z = 0$.

A convenient and useful transformation for three-dimensional boundary layer flows is given in ref. 15. According to this transformation, which is a generalization of 2-d Falkner-Skan transformation [1] to 3-d flows, we first define the transformed coordinates by

$$x = x, \quad z = z \quad d\eta = \left(\frac{u_e}{\rho_e \mu_e s_1} \right)^{1/2} \rho dy, \quad s_1 = \int_0^x h_1 dx \quad (24)$$

and introduce a two-component vector potential such that

$$\begin{aligned} \rho u h_2 \sin \theta &= \frac{\partial \psi}{\partial y}, & \rho w h_1 \sin \theta &= \frac{\partial \phi}{\partial y} \\ \overline{\rho v} h_1 h_2 \sin \theta &= - \left(\frac{\partial \psi}{\partial x} + \frac{\partial \phi}{\partial z} \right) \end{aligned} \quad (25)$$

In addition, we define dimensionless ψ and ϕ by

$$\begin{aligned} \psi &= (\rho_e \mu_e u_e s_1)^{1/2} h_2 \sin \theta f(x, z, \eta) \\ \phi &= [(\rho_e \mu_e u_e s_1)^{1/2} u_{ref} / u_e] h_1 \sin \theta g(x, z, \eta) \end{aligned} \quad (26)$$

Using these transformations and the concepts of eddy viscosity and turbulent Prandtl number it can be shown that the x, z -momentum equations and the equation and their boundary conditions in transformed variables can be written as:

x -Momentum

$$\begin{aligned} (bf'')' + m_1 f f'' - m_2 (f')^2 - m_5 f' g' + m_6 f'' g - m_8 (g')^2 + m_{11} c \\ = m_{10} \left(f' \frac{\partial f'}{\partial x} - f'' \frac{\partial f}{\partial x} \right) + m_7 \left(g' \frac{\partial f'}{\partial z} - f'' \frac{\partial g}{\partial z} \right) \end{aligned} \quad (27)$$

z -Momentum

$$\begin{aligned} (bg'')' + m_1 f g'' - m_4 f' g' - m_3 (g')^2 + m_6 g g'' - m_9 (f')^2 + m_{12} c \\ = m_{10} \left(f' \frac{\partial g'}{\partial x} - g'' \frac{\partial f}{\partial x} \right) + m_7 \left(g' \frac{\partial g'}{\partial z} - g'' \frac{\partial g}{\partial z} \right) \end{aligned} \quad (28)$$

Energy

$$(\mu_1 E')' + \mu_2 E' + \mu_3' = m_{10} \left(f' \frac{\partial E}{\partial x} - E' \frac{\partial f}{\partial x} \right) + m_7 \left(g' \frac{\partial E}{\partial z} - E' \frac{\partial g}{\partial z} \right) \quad (29)$$

Boundary Conditions:

$$\begin{aligned} \eta = 0 : \quad & f_w = f'_w = g_w = g'_w = 0, \quad E'_w = 0 \\ \eta = \eta_\infty : \quad & f' = 1, \quad g' = \frac{w_e}{u_{ref}}, \quad E = 1 \end{aligned} \quad (30)$$

Here primes denote differentiation with respect to η and

$$f' = u/u_e, \quad g' = w/u_{ref}, \quad E = H/H_e \quad (31)$$

$$b = C(1 + \epsilon_m^+), \quad C = \frac{\rho u}{\rho_e u_e}, \quad c = \frac{\rho_e}{\rho}, \quad \epsilon_m^+ = \epsilon_m / \nu$$

The coefficients m_1 to m_{12} are functions of external velocity distribution, the coordinate system and fluid properties as described in ref. 15. The formulas for ϵ_m^+ are also described in ref. 15.

To solve the system given by (27) to (30) by the Box method, we use the net cube shown in Figure 5. Here the equations are centered by the midpoint of the cube (see ref. 16). With the solution obtained along the stagnation line at the

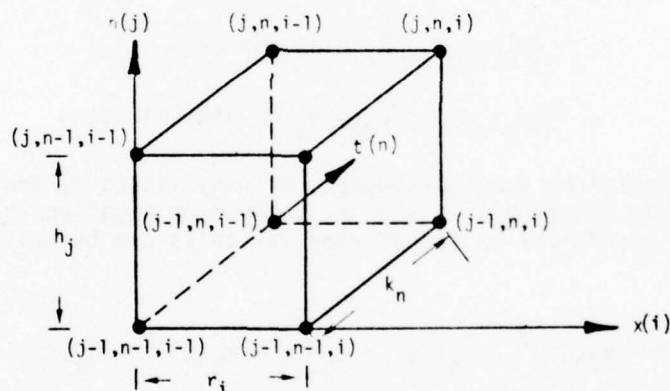


Figure 5. The net cube for three-dimensional flows.

leading edge, all that remains is the specification of the integration direction in z . This cannot be done arbitrarily by stating that the integration will always begin at the wing root, after the solution of some appropriate equations there, and proceed towards the tip. It is possible, and, in fact, we have cases where it was required, that the integration proceeds from tip to root [15]. I will return to the details of this very important point later.

For the initial work we did, using the cube shown in Figure 5, the sign of the external velocity, w_e , was used to determine the direction of integration. For $w_e > 0$ the procedure integrates from root to tip, however, if $w_e < 0$, the integration direction is reversed. The starting solution in the plane of the tip is simply an approximation to the true equations there. One test case we calculated had three regions along the wing, see Figure 6.

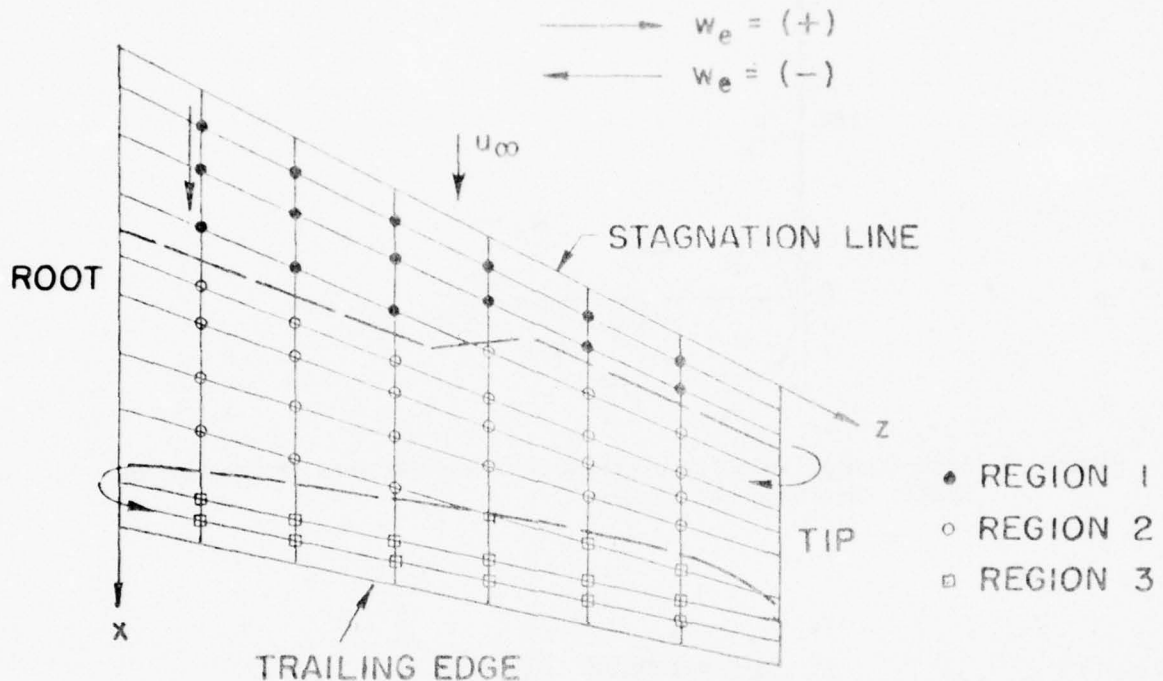


Figure 6. Definitions of various regions on the wing used for marching procedure.

As shown in this figure the direction of integration is out from root to tip, then in, and finally out again. Using this very complicated logic, for the external velocity shown in Figure 7, we obtained the solution of the governing equations without any numerical difficulties. Figure 8 shows a comparison of calculated and experimental results and Figure 9 shows the computed cross-flow velocity profiles for this case. Since this particular wing was of a special nature, the three-dimensional solution could be compared with an infinite swept-wing solution obtained using an orthogonal system by simply using a coordinate rotation. The agreement was excellent.

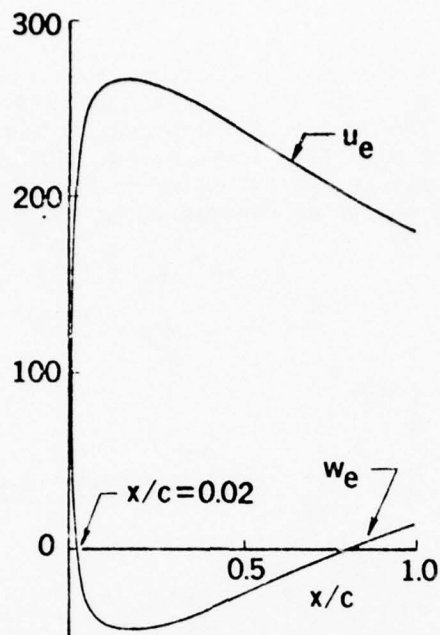


Figure 7. Experimental velocity distribution for the data of Brebner and Wyatt; nonorthogonal system.

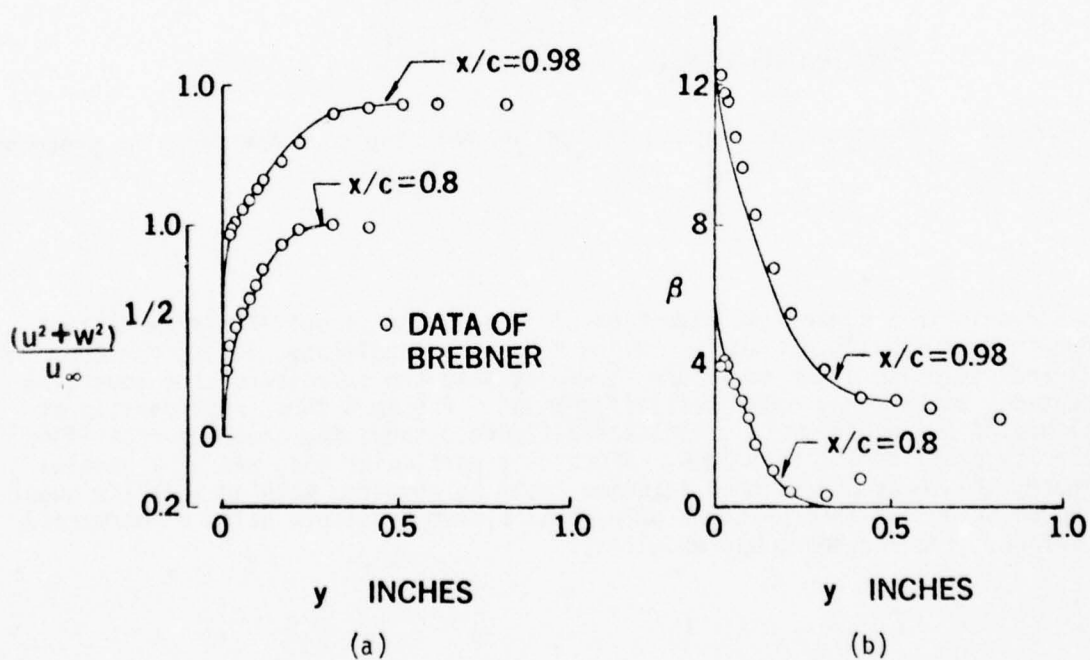


Figure 8. Comparison of calculated results with experiment for the data of Brebner and Wyatt. (a) Velocity profiles. (b) Cross-flow-angle distribution.

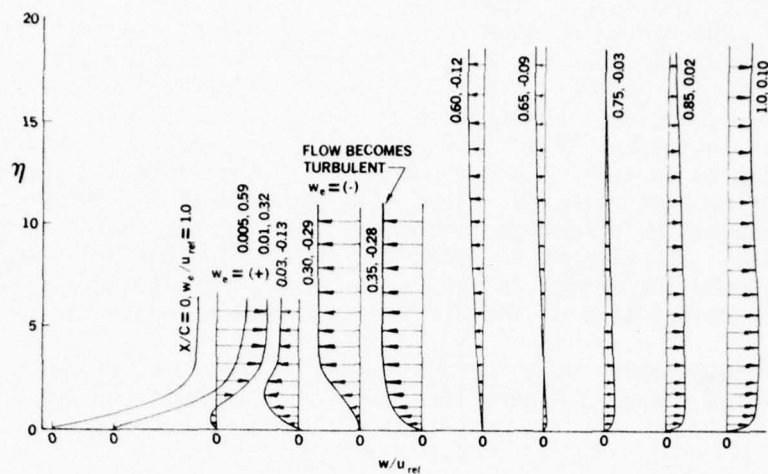


Figure 9. Computed cross-flow velocity profiles at different chordwise stations for the data of Brebner and Wyatt, nonorthogonal system, $u_{ref} = w_e$ at $x = 0$.

Numerous other test cases for wings were performed, including one for which an isolated "island" of $w_e < 0$ existed on the wing, see Figure 10,

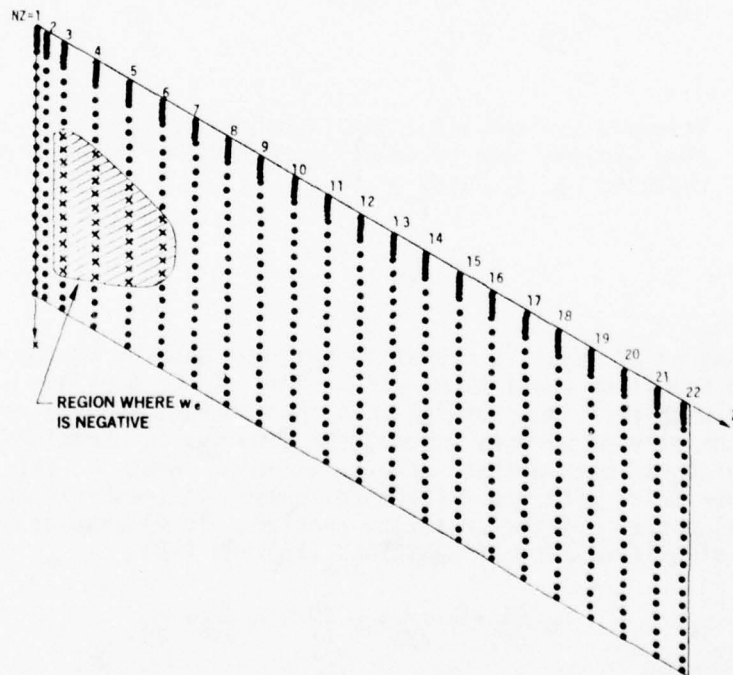


Figure 10. Finite wing. The symbols denote the stations where the boundary-layer calculations are made. Dots correspond to stations where w_e is positive and x's correspond to negative w_e .

not connected to the root or the tip. Our procedure still covered the entire wing surface, and generated good results. Nonetheless, we felt that our method of determining the z integration direction was inexact, and continued to seek better alternatives. This was further stimulated by our work on another three-dimensional boundary-layer problem, the flow past a body of revolution at incidence [16]. This configuration produces a region of reverse crossflow, due to an adverse pressure gradient, but without a change of sign of the crossflow edge velocity. Thus, the method used on the wing, where the crossflow integration direction is changed when the edge velocity changes sign, cannot be used. The attempt to integrate into an increasingly negative flow ultimately caused the method to break down apparently due to a violation of the stability restriction of the difference scheme shown in Figure 5.

A possible solution to this crossflow problem can be found by the use of the differencing shown in Figure 11, called zig-zag differencing. This allows

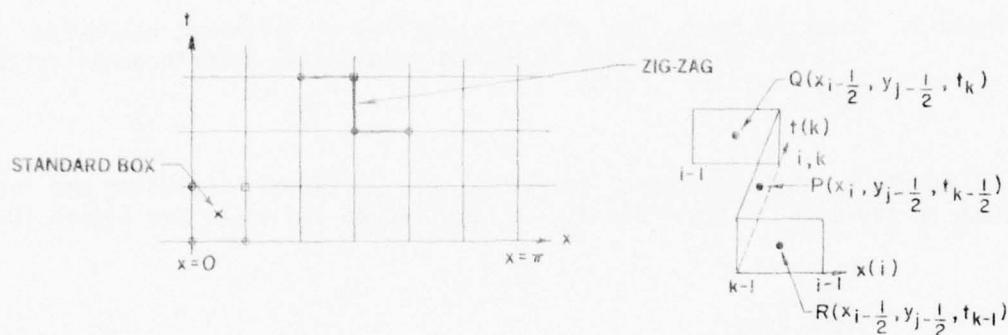


Figure 11. Standard box and zig-zag differencing schemes for the unsteady flow problem used by Cebeci [18] for 3-D steady flows, t is replaced by x , and x by z .

a greater degree of negative crossflow than our cube by virtue of a less restrictive stability requirement [17]. Some of the previous wing calculations were redone with this incorporated into the program, and the results were basically the same as before, except that some small, unnoticed ripples in the solution which had been present were eliminated. However, this seemed to be just a clever trick with the difference scheme used, and did not really address the physical process of the crossflow problem. To illustrate our current method, consider a simplified form of equations (17) and (18).

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} = -\frac{\partial p}{\partial x} + \frac{\partial \tau}{\partial y} \quad (32a)$$

$$u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} = - \frac{\partial p}{\partial z} + \frac{\partial \tau}{\partial z} \quad (32b)$$

The problem is that u and w continuously change through the boundary layer, as well as across the wing at the boundary-layer edge; the normal velocity, v , does not concern us here. Somehow these changes must be taken into account so that we always march in a direction for which the diffusion problem is well posed. With this in mind, we can look at the equations as expressing the variation of quantities along characteristic directions:

$$u \frac{\partial u}{\partial x} + w \frac{\partial u}{\partial z} \equiv \frac{\partial u}{\partial s} = - \frac{\partial p}{\partial x} + \frac{\partial \tau}{\partial y} - v \frac{\partial u}{\partial y} \quad (33a)$$

$$u \frac{\partial w}{\partial x} + w \frac{\partial w}{\partial z} \equiv \frac{\partial w}{\partial s} = - \frac{\partial p}{\partial z} + \frac{\partial \tau}{\partial z} - v \frac{\partial w}{\partial z} \quad (33b)$$

We approximate the equations by differencing backward along the characteristic direction which exists at each different y -level, see Figure 12. Thus, this

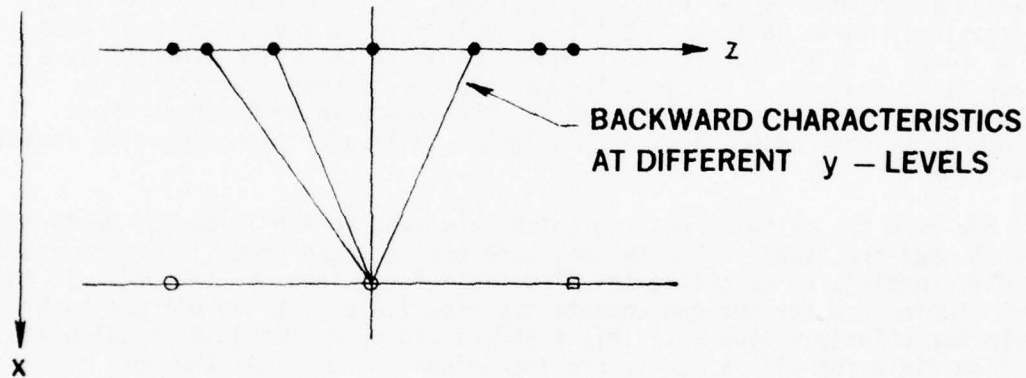


Figure 12. The new differencing scheme for three-dimensional flows. The closed symbols denote the computed solutions. The marching procedure is in the spanwise direction for a given chordwise station.

method uses the notion of domains of dependence more carefully, and follows the characteristics of the locally plane flow. Our calculations using this method on the wing again have reproduced our original calculations without any oscillations, and have greatly reduced the logic of the program allowing the calculation to proceed from root to tip at every x -station. We feel that this scheme represents the most realistic and accurate way to compute three-dimensional boundary layers on arbitrary wings.

Another area where these same considerations arise is in unsteady two-dimensional boundary layers. The governing equations for laminar flow are:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (34)$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = \frac{\partial u_e}{\partial t} + u_e \frac{\partial u_e}{\partial x} + v \frac{\partial^2 u}{\partial y^2} \quad (35)$$

They are subject to the boundary conditions

$$y = 0: \quad u = v = 0; \quad y = \delta \quad u = u_e(x, t) \quad (36)$$

and an appropriate initial condition.

I will illustrate the properties of these types of flows by considering the flow over a circular cylinder started impulsively from rest [18]. In this flow at certain times, the streamwise velocity profile contains regions of backflow. Thus, methods used for two-dimensional separated flows, and three-dimensional flows with reverse crossflow should be applicable here. The flow development proceeds as follows. For some time after the impulsive start at $t = 0$, all the flow is from forward to rear stagnation point, and so the skin friction, c_f , is positive. At time $t \approx 0.32$, c_f becomes negative at an interior point on the cylinder, and then this $c_f = 0$ point progressively moves upstream until it finally reaches a value of $x \approx 105^\circ$ at $t = 1.25$. This value of x is quite close to the value computed by a steady-state analysis. However, the calculation assumes the flow to be unsteady, and models the vortex shedding phenomenon in some sense. Here the boundary-layer assumptions are no longer valid and the calculation finally breaks down.

How were the calculations, on which this description is based, performed? I first used the standard Box method, with the net cube shown in Figure 5 used for differencing, to calculate the flow field for values of $t < 0.8$. In this way I studied how far one can compute the flow field with and without backflow before the solutions blow up. This standard procedure was able to calculate the flow field for all x up to and including $t = 0.6$. At the next time interval no trouble was encountered up to $x \approx 152^\circ$. Although convergence was obtained at the next two x -stations, the asymptotic behavior at the edge of the boundary layer was not correct. Finally, at $x \approx 156^\circ$ the solutions diverged. With increasing time, the last "good" station moved forward so that it was at $x \approx 136^\circ$ at $t = 0.8$. Next I used the zig-zag scheme of Figure 11 to compute the flow. For all practical purposes, the results obtained earlier by the standard Box scheme agreed with those obtained by the new procedure except now the solutions did not break down at those x -stations previously mentioned, and the calculations were performed up to $x = 180^\circ$ for values of t up to 0.80. For values of $t > 0.80$, the solutions began to develop oscillations in regions of backflow. To remedy this, I used the procedure described as global iteration in the separated flow section, and made two passes in x for each time level, differencing with the wind in regions of backflow. In this way the calculations were performed until $t = 1.25$, and values of skin friction and displacement thickness as well as velocity profiles were computed without any difficulty. Figure 13 shows the computed skin friction and displacement thickness distributions around the circular cylinder for various values of t . I am presently investigating better procedures for this flow in order to extend the calculations to large times, hopefully to $t = \infty$. The results will be reported in a forthcoming report.

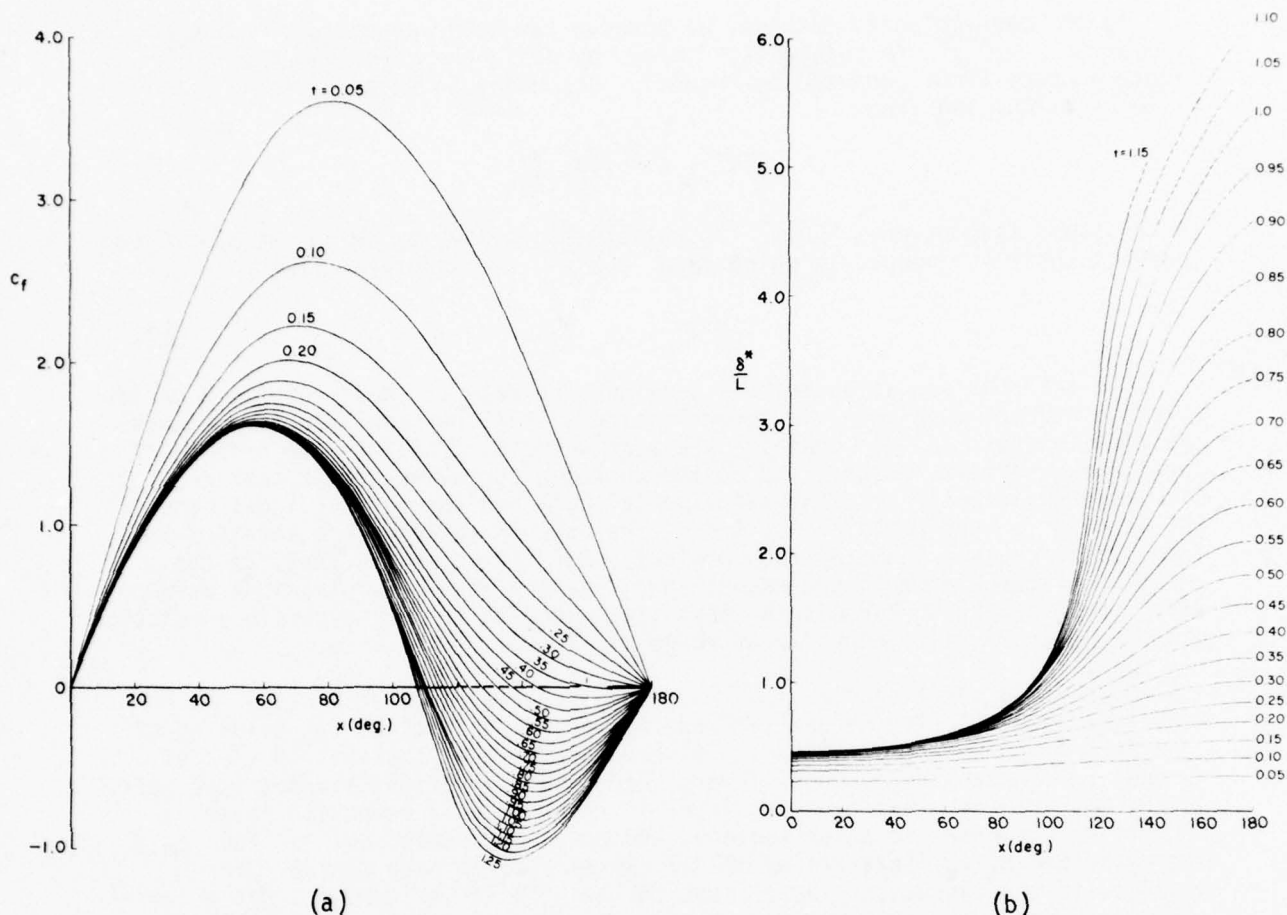


Figure 13. Computed: (a) Local skin-friction coefficients, and (b) Displacement thickness distributions for the unsteady flow problem discussed in [18].

4. TURBULENCE MODELLING. I would like to conclude with a brief summary of the types of models being used in turbulent-flow calculations. In general, the prediction methods for turbulent boundary layers can be divided into integral and differential methods. The integral methods involve the integral parameters of the boundary layer (e.g., displacement thickness, local skin friction, shape factor, etc.). They avoid the complexity of solving the boundary-layer equations in the partial-differential-equation form and instead solve a system of ordinary differential equations. For further details, see references 19, 20.

Differential methods involve direct assumptions for the shear stress $-\rho \bar{u}'v'$ and seek the solution of the governing equations in their partial-differential-equation form. The assumption can either lead to an algebraic relation between $-\rho \bar{u}'v'$ and $\partial u/\partial y$ or it can take the form of a partial-differential equation. The methods that use the first assumption are often called mean-velocity methods; those that use the second assumption are called transport-equation methods.

In the mean-velocity methods, we observe two main assumptions. In one assumption, $-\rho \overline{u'v'}$ is related to $\partial u / \partial y$ by using the so-called mixing-length concept first proposed by Prandtl. According to that concept, $-\rho \overline{u'v'}$ is to be calculated from

$$-\rho \overline{u'v'} = \rho \ell^2 \left| \frac{\partial u}{\partial y} \right| \frac{\partial u}{\partial y} \quad (37)$$

In the other assumption, $-\rho \overline{u'v'}$ is related to $\partial u / \partial y$ by using the so-called "eddy-viscosity" concept, in which case $-\rho \overline{u'v'}$ is calculated from

$$-\rho \overline{u'v'} = \rho \epsilon_m \frac{\partial u}{\partial y} \quad (38)$$

The transport-equation methods consider the rate of change of the Reynolds stresses and are more accurate general methods than the mean velocity methods. The equation for $-\rho \overline{u'v'}$ contains the mean velocity gradient, one or more of the unknown normal stresses, and further unknown turbulence terms that can be simulated empirically by using experimental data and theoretical ideas about the behavior of the turbulence. Again, the unknown turbulence quantities are found empirically. That approach obviously has the potential that, as the behavior of turbulence is understood more, one can make more plausible assumptions for the unknown turbulence quantities, leading to more accurate prediction methods that are valid for a large range of flows.

Both the assumption for $-\rho \overline{u'v'}$ given either by (37) or by (38) can be used in a mean-velocity method such as the one which involves the solution of equations given by (1) to (3). It is necessary to know the distribution of ℓ and ϵ_m across the boundary layer. According to various studies (see refs. 1, 19, 20) the turbulent boundary layer is regarded as a composite layer consisting of inner and outer regions, and the distributions of ℓ and ϵ_m are described by two separate empirical expressions in each region. For example, if the viscous sublayer close to the wall is excluded, ℓ is proportional to y in the inner region, and it is proportional to δ in the outer region. Therefore,

$$\ell = \begin{cases} \kappa y & y_0 \leq y \leq y_c \\ \alpha_1 \delta & y_c \leq y \leq y_e \end{cases} \quad (39a)$$

$$(39b)$$

Here y_0 is a small distance from the wall, a distance approximately equal to $60 \nu (\tau_w / \rho)^{-1/2}$, and y_c is another distance obtained from the continuity of mixing length. The empirical parameters κ and α_1 vary slightly according to experimental data. For flows at high Reynolds numbers ($R_\theta > 5000$), they are generally taken to be $\kappa = 0.40$ and $\alpha_1 = 0.075$. For further details, see ref. 19.

Similarly, according to various studies, ϵ_m varies linearly with y in the inner region and is nearly constant in the outer region. Its variation across the boundary layer can conveniently be described by the following formulas:

$$\epsilon_m = \begin{cases} \ell^2 \left| \frac{\partial u}{\partial y} \right| & y_0 \leq y \leq y_c \\ \alpha \int_0^\infty (u_e - u) dy & y_c \leq y \leq \delta \end{cases} \quad (40a)$$

$$(40b)$$

with ℓ given by (39a). The parameter α is generally assumed to be a universal constant equal to 0.0168 for $R_\theta > 5000$.

There have been numerous studies on extending (39) and (40) to include the viscous sublayer. By an analogy with the laminar flow on an oscillating plate, Van Driest suggested modifying (39a) to

$$\ell = \kappa y [1 - \exp(-y/A)] \quad (41)$$

for a smooth flat-plate flow. Here A is a damping-length constant, for which, the best dimensionally correct empirical choice is about $A^+ \nu (\tau_w/\rho)^{-1/2}$ with A^+ denoting an empirical constant equal to 26.

Needless to say, the eddy viscosity and mixing-length formulas, like most (if not all) expressions for turbulent flows, are empirical. Over the years, several empirical corrections to these formulas have been made, to account for the effects of low Reynolds number, transitional region, compressibility, mass transfer, pressure gradient, and transverse curvature. See the discussion in ref. 19 for complete details.

The transport equation methods, in general, use two different approaches to model the Reynolds shear stress $-\rho \overline{u'v'}$. These are Two-Variable Models, and Reynolds Stress Models. Here we shall discuss only the more commonly used one, namely the Two-Variable Model. Methods based on two-variable models use the turbulent energy equation, which for two-dimensional incompressible turbulent flows is

$$u \frac{\partial}{\partial x} \left(\frac{q^2}{2} \right) + v \frac{\partial}{\partial y} \left(\frac{q^2}{2} \right) = - \frac{\partial}{\partial y} \left(\frac{\overline{pv'}}{\rho} + \frac{\overline{v'q^2}}{2} \right) - \overline{u'v'} \frac{\partial u}{\partial y} + \nu \frac{\partial^2}{\partial y^2} \left(\frac{q^2}{2} \right) - \phi \quad (42)$$

In general, the methods consist of two types. One type uses the value of q from (42) to form an eddy viscosity ϵ_m . Since eddy viscosity is the product of a velocity and length,

$$\epsilon_m \sim \text{velocity} \times \text{length}$$

and these methods define ϵ_m by

$$\epsilon_m = c_1 q \ell \quad (43)$$

Here ℓ is a turbulence length scale, and c_1 is a constant at high Reynolds number of turbulence, $R_t = q\ell/\nu$. In the current calculation methods, ℓ is specified algebraically or through a differential equation. When ℓ is specified algebraically, the prediction method requires the solution of three partial-differential equations, (2), (3), and (42) with suitable assumptions for the unknown turbulence quantities appearing in (42). When ℓ is specified by a differential equation, the prediction method requires the solution of four partial-differential equations, (2), (3), (42), and a length-scale equation.

A second type of method, advocated by Bradshaw and his associates, see, for example, refs. 21 and 22, uses the turbulent energy equation to form a relation for the Reynolds shear stress. Bradshaw relates the Reynolds shear stress to q^2 by

$$-\overline{u'v'} = a_1 q^2 \quad (44)$$

Here a_1 is a universal constant assumed to be 0.15.

Both types require closure assumptions for the unknown turbulence quantities appearing in (42). However, the assumptions for those quantities do not differ from one type to another, once the decision of how to utilize the turbulent kinetic energy equation is made. For example, the eddy-viscosity methods model the production term as

$$-\overline{u'v'} \frac{\partial u}{\partial y} = \epsilon_m \left(\frac{\partial u}{\partial y} \right)^2 \quad (45)$$

The Bradshaw-type methods model it as

$$-\overline{u'v'} \frac{\partial u}{\partial y} = a_1 q^2 \frac{\partial u}{\partial y} \quad (46)$$

Both approaches model the dissipation term as

$$\phi = c_2 \frac{q^3}{\ell} \quad (47)$$

A practical but not necessary difference between the two approaches comes in modeling the "diffusion" term. Eddy-viscosity methods model it by relating it to the gradient of q^2 in the form

$$-\left(\frac{\overline{pv'}}{\rho} + \frac{\overline{v'q^2}}{2} \right) = c_3 \epsilon_m \frac{\partial}{\partial y} \left(\frac{q^2}{2} \right) \quad (48)$$

where c_3 is a constant or a specified function. The Bradshaw-type methods model it as

$$\frac{\overline{pv'}}{\rho} + \frac{\overline{v'q^2}}{2} = G Q q^2 \quad (49)$$

Here G is a constant or a specified function, and Q is a velocity scale characteristic of the large eddy motions.

It should be noted that the closure assumption for the diffusion term is of considerable importance. With the assumption in (48), equation (42) is parabolic; and with (49), equation (42) is hyperbolic. Of the two-variable methods, Bradshaw's method has been extensively used for a wide range of flow conditions by himself and by others, see ref. [20].

As an example of the complex problems which can be solved using the relatively simple mean flow models, we have computed the entry flow in a pipe, see figure 14. Here there are two distinct regions of the flow, with an

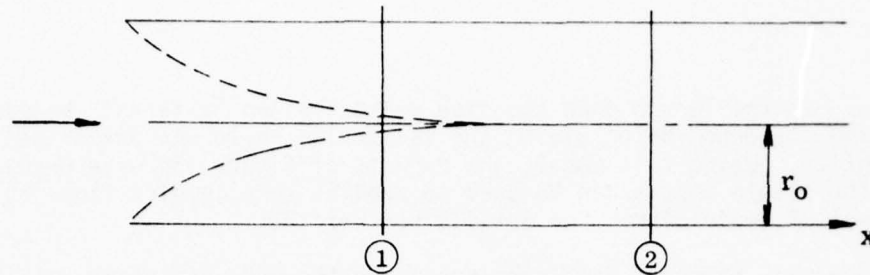


Figure 15. Entry flow problem in a circular pipe. ① denotes the approximate location where shear layers merge. ② denotes the region where the flow is fully developed.

overlapping zone between them. Near the entrance of the pipe the flow behaves like an external flow, $x < x_1$, and after some entry length, it becomes fully developed, $x > x_2$, and has all the attributes of an internal flow. It is known that the eddy-viscosity formulations in both regions are totally different.

If the flow development is computed using either one of the two eddy viscosities separately, and compared with experimental data, the result is as shown in Figure 15a. The eddy viscosity formulation used for external

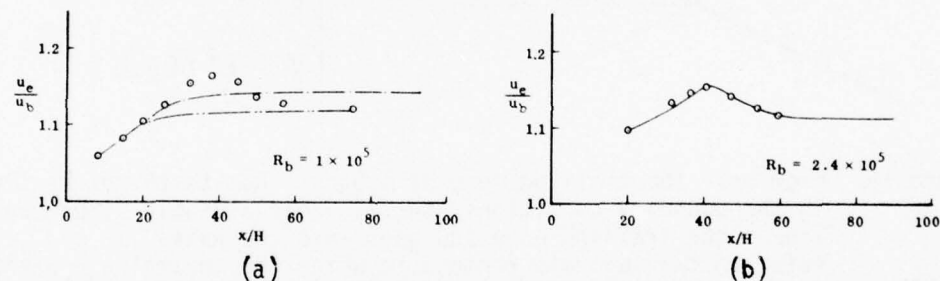


Figure 15. Comparison of calculated and experimental centerline velocity distributions for the entry flow problem using three different eddy viscosity formulations. (a) — · — denotes results obtained by (40), — · — denotes results obtained by using a mixing-length distribution for fully developed flows. (b) — denotes results obtained by (50). The data is due to Dean.

flows produces results that are in good agreement with experiment for $x < x_1$, and the mixing length formulation used for fully developed internal flows (see ref. 1, for example) produces results that are in good agreement with experiment for $x > x_2$. However, the overall agreement is poor. Our solution was to model the entire flow situation based on our knowledge of the physics. We prescribe a composite eddy viscosity which changes smoothly from an external (ϵ_1) to an internal (ϵ_2) model as

$$\epsilon_m = \epsilon_1 + (\epsilon_2 - \epsilon_1) \left[1 - \exp - \left(\frac{x - x_0}{20} \right) \right] \quad (50)$$

where x_0 is the length down the pipe where the two "external" boundary layers merge, and the denominator giving the relaxation scale was prescribed by divine intervention. Using this model, the results of Figure 15b were produced. Hence, the simple models can be used to predict some complex flows by using a great deal of empiricism. For details see ref. 23.

There are, however, some problems where the eddy-viscosity, mixing-length formulations may not work. One example is a turbulent flow near the trailing edge of bodies (Figure 16) where the multiple structure within the boundary layer imposes additional scales on the turbulence field which may not be handled by simple models. Another example occurs in separating and reversed flows. In reverse-flow regions, the velocity profile (see Figure 1) must have $\partial u / \partial y \equiv 0$ within the $u < 0$ zone. According to simple eddy-viscosity models, i.e. eq. (40a), the turbulent shear stress vanishes at this point. This is known to be an unrealistic result and so turbulent separating flows are an area where both the mathematics and the physics of the problem need more investigation.

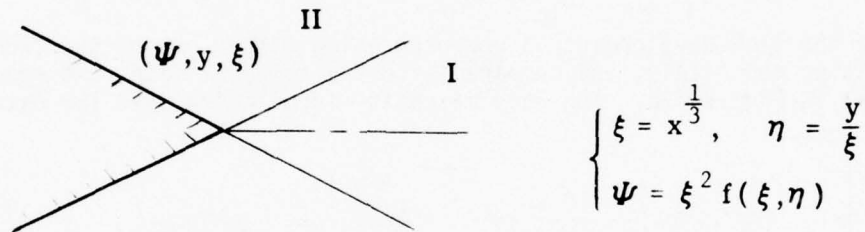


Figure 16. Flow near the trailing edge of a body. Due to the discontinuity in the boundary conditions, the flow has a double structure as it leaves the trailing edge and goes into the wake. I and II refer to two separate regions in which the governing equations are expressed in different scaling variables.

5. ACKNOWLEDGMENT. This work was partially supported by the Office of Naval Research under Contract N00014-77-C-0073.

REFERENCES

1. Cebeci, T. and Bradshaw, P. (1977). Momentum Transfer in Boundary Layers. Hemisphere-McGraw Hill, Washington.
2. Keller, H.B. (1970). A new difference scheme for parabolic problems. Numerical Solutions of Partial Differential Equations. II, J. Bramble (ed.) Academic Press, New York.

3. Keller, H.B. and Cebeci, T. (1971). Accurate numerical methods for boundary-layer flows. I. Two-dimensional laminar flows. Lecture Notes in Physics, 8, Proceedings of the Second International Conf. on Numerical Methods in Fluid Dynamics, p. 92, Springer-Verlag, Berlin and New York.
4. Keller, H.B. and Cebeci, T. (1972). Accurate numerical methods for boundary-layer flows. II. Two-dimensional turbulent flows. AIAA J., 10, 1193.
5. Keller, H.B. and Cebeci, T. (1972). An inverse problem in boundary-layer flows: numerical determination of pressure gradient for a given wall shear. J. Comp. Physics, 10, 151.
6. Cebeci, T. and Keller, H.B. (1973). Laminar boundary layers with assigned wall shear. Lecture Notes in Physics (Proc. Int. Conf. Numer. Methods Fluid Dynam., 3rd), 19, Springer-Verlag, Berlin and New York.
7. Brown, S.N. and Stewartson, K. (1969). Laminar separation. Annu. Rev. Fluid Mech., 1, p. 45-72.
8. Catherall, D. and Mangler, R.W. (1966). The integration of the two-dimensional laminar boundary-layer equations past the point of vanishing skin friction. J. Fluid Mech., 26 (1), p. 163-182.
9. Cebeci, T. (1976). Separated flows and their representation by boundary-layer equation. Mech. Engg. Rept. No. ONR-CR215-234-2, California State University at Long Beach.
10. Cebeci, T. (1976). An inverse boundary-layer method for compressible laminar and turbulent flows. J. of Aircraft, 13, p. 709.
11. Reyhner, T.A. and Flugge-Lotz, I. (1968). The interaction of a shock wave with a laminar boundary layer. Int. J. Non-Linear Mech., 3 (2), p. 173-199.
12. Williams, P.G. (1975). A reverse flow computation in the theory of self-induced separation. Lecture Notes in Physics (Proc. Inter. Conf. Numer. Methods Fluid Dynam., 4th), 35, Richtmyer, R.D. (ed), Springer-Verlag, p. 445-451.
13. Briley, W.R. (1971). A numerical study of laminar separation bubbles using the Navier-Stokes equations. J. Fluid Mech., 47 (4), p. 713-736.
14. Carter, J.E. (1975). Inverse solutions for laminar boundary-layer flows with separation and reattachment. NASA TR R-447.
15. Cebeci, T., Kaups, K., and Ramsey, J.A. (1977). A general method for calculating three-dimensional compressible laminar and turbulent boundary layers on arbitrary wings. NASA CR 2777.

16. Hirsh, R.S. and Cebeci, T. (1977). Calculation of three-dimensional boundary layers with negative cross-flow on bodies of revolution. AIAA Paper No. 77-683.
17. Krause, E., Hirschel, E.H. and Bothmann, Th. (1968). Die numerische integration der bewegungsgleichungen driedimensionaler laminarer kompressibler grenzschichten. Fachtagung Aerodynamik, Berlin, D6LR-Fachlinchreihe, Bond 3.
18. Cebeci, T. (1977). On the solution of laminar flow over a circular cylinder started impulsively from rest, in "Professor A. Walz' 70th Birthday Anniversary Volume," to be published, 1977b.
- 19., Cebeci, T. and Smith, A.M.O. (1974). Analysis of Turbulent Boundary Layers Applied Mathematics and Mechanics, 15, Academic Press, New York.
20. Bradshaw, P. (ed.) (1976). Topics in Applied Physics, Vol. 12 - Turbulence. Springer, Heidelberg.
21. Bradshaw, P. and Ferriss, D.H. and Atwell, N.P. (1967). Calculation of boundary layer development using the turbulent energy equation. J. Fluid Mech., 28, p. 593.
22. Bradshaw, P. and Ferriss, D.H. (1972). Applications of a general method of calculating turbulent shear layers. J. Basic Eng., 94D, p. 345.
23. Cebeci, T. and Chang, K.C. (1977). A general method for calculating momentum and heat transfer in laminar and turbulent duct flows. To be published in Numerical Heat Transfer.

APPROXIMATION WITH VPB SPLINES

Royce W. Soanes, Jr.
US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND
Large Caliber Weapon System Laboratory
Benet Weapons Laboratory
Watervliet, N. Y. 12189

ABSTRACT. This article makes an extension of the interpolating VP (variable power) splines defined in [6, 7] to define approximating VPB (variable power basic) splines. VPB splines are of a more general nature than VP splines because they may be conveniently used for linear approximation in addition to interpolation. VPB splines will also reduce to C^2 cubic B splines as a special case.

1. INTRODUCTION. The variable power basic (VPB) splines introduced here are an outgrowth of the ideas contained in [6, 7] with an assist from the very readable book by Prenter [4]. The articles on VP splines were written with the objective of providing and making use of a relatively simple interpolant to which one could apply a certain amount of local control without losing C^2 smoothness.

B splines have been used extensively for solving linear approximation problems [1, 2], but in order to make the best use of B splines in least squares curve fitting, one must consider the nonlinear problem of optimal knot placement [3, 5]. This problem can involve a considerable amount of computation if the number of knots is large. For this reason, we will consider only fixed knots here. Although the knots will be fixed, we will attempt to select them in an intuitively reasonable manner which, in addition to the setting of the nonlinear parameters of the VPB splines, will afford us fairly appealing data fits.

2. SOME BASIC VP SPLINE FORMULAS. The tridiagonal linear system of equations producing second derivative continuity of a VP spline interpolant is given by:

$$(m_1 - 1)y'_1 + y'_2 = m_1 q_1$$

$$A_i y'_{i-1} + B_i y'_i + C_i y'_{i+1} = D_i q_{i-1} + E_i q_i \quad (1 < i < N)$$

$$y'_{N-1} + (n_{N-1} - 1)y'_N = n_{N-1} q_{N-1}$$

where

$$A_i = m_{i-1}(m_{i-1} - 1)/(k_{i-1} l_{i-1})$$

$$C_i = n_i(n_i - 1)/(k_i \ell_i)$$

$$B_i = (n_{i-1} - 1)A_i + (m_i - 1)C_i$$

$$D_i = n_{i-1} A_i$$

$$E_i = m_i C_i$$

and

$$\ell_i = x_{i+1} - x_i$$

$$q_i = (y_{i+1} - y_i)/\ell_i$$

$$k_i = m_i + n_i - m_i n_i$$

The nonlinear parameter vectors m and n may be set as in [6, 7] after the knots have been selected and local smoothing has been applied to produce corresponding y values. A rule for setting m and n is given by:

$$R_i = n_i/m_{i-1} = (\ell_i/\ell_{i-1})^2(S_{i-1}/S_i)$$

$$m_{i-1} = L \text{ and } n_i = LR_i \text{ if } R_i \geq 1$$

$$n_i = L \text{ and } m_{i-1} = L/R_i \text{ if } R_i < 1$$

$$\text{for } 2 \leq i \leq N-1$$

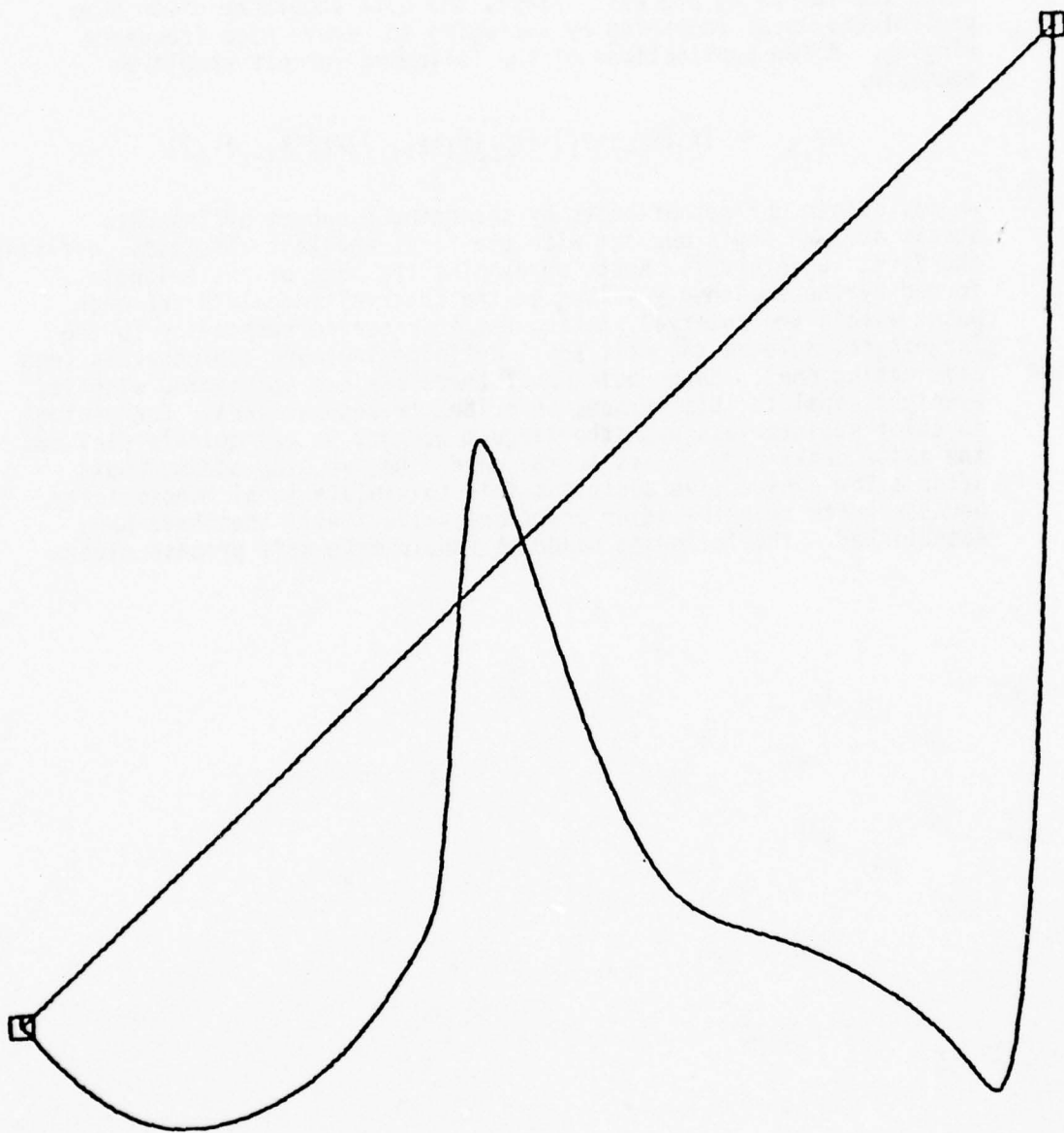
$$n_1 = L \text{ and } m_{N-1} = L$$

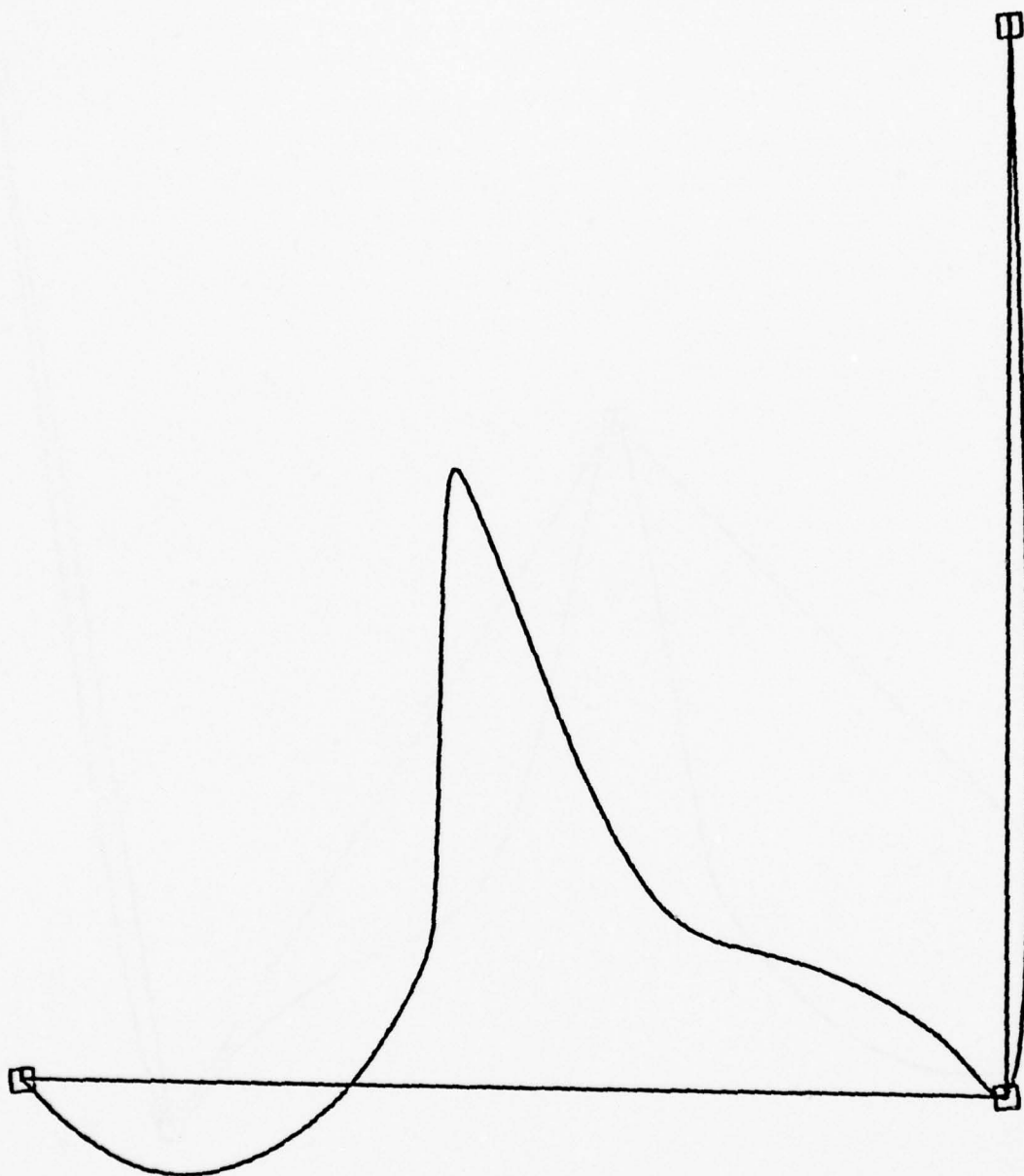
where L (>2) is a lower bound on all m_i and n_i . S_i is the chord length on the i th knot subinterval.

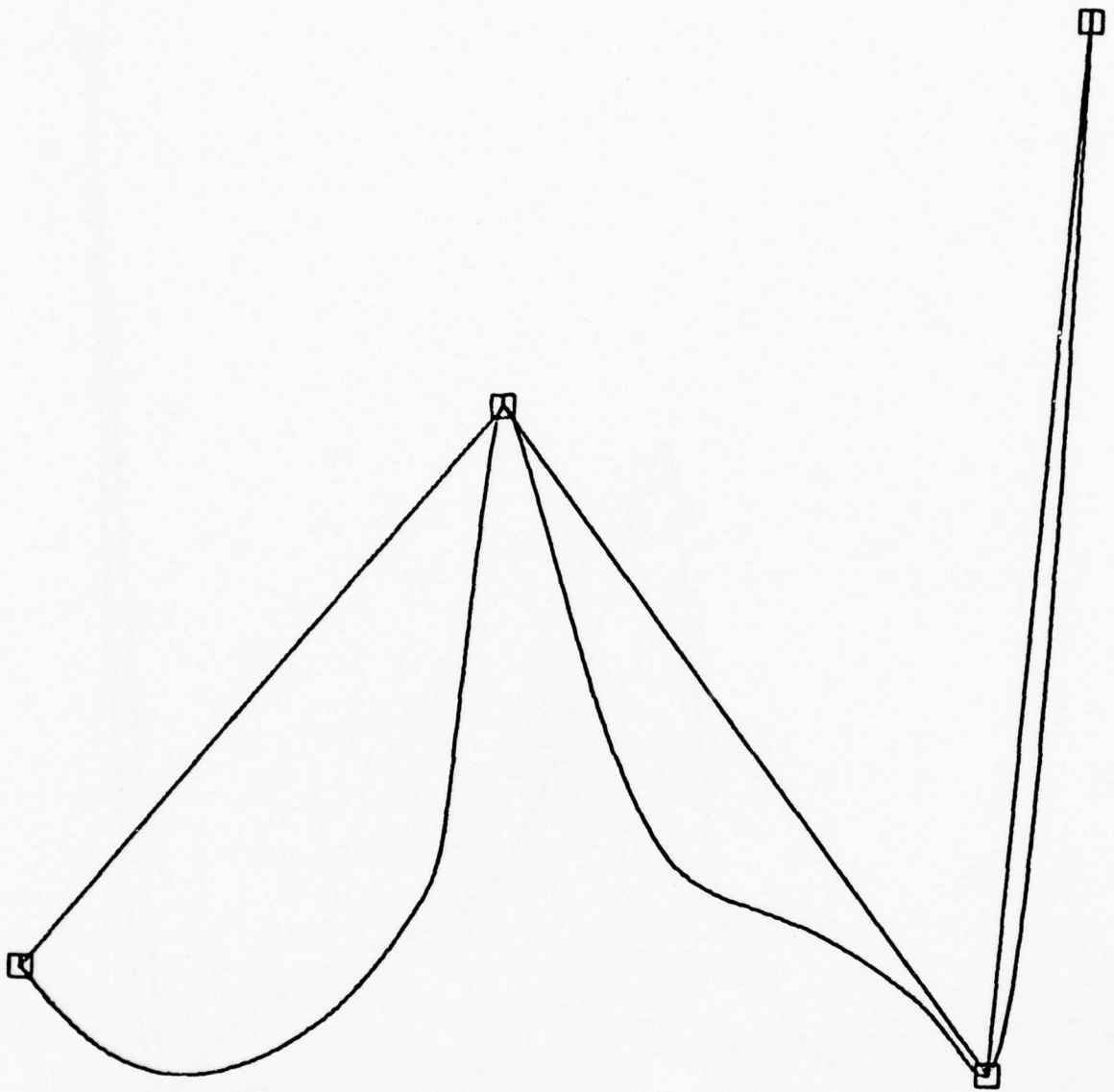
3. KNOT SELECTION. A fairly good set of knots may be selected using the following process. First, the data should be given some preliminary local smoothing by averaging to remove high frequency ringing. A few applications of the following formula should be adequate.

$$y_{i,S} = [\ell_i(y_{i-1}+y_i) + \ell_{i-1}(y_i+y_{i+1})]/[2(\ell_{i-1}+\ell_i)]$$

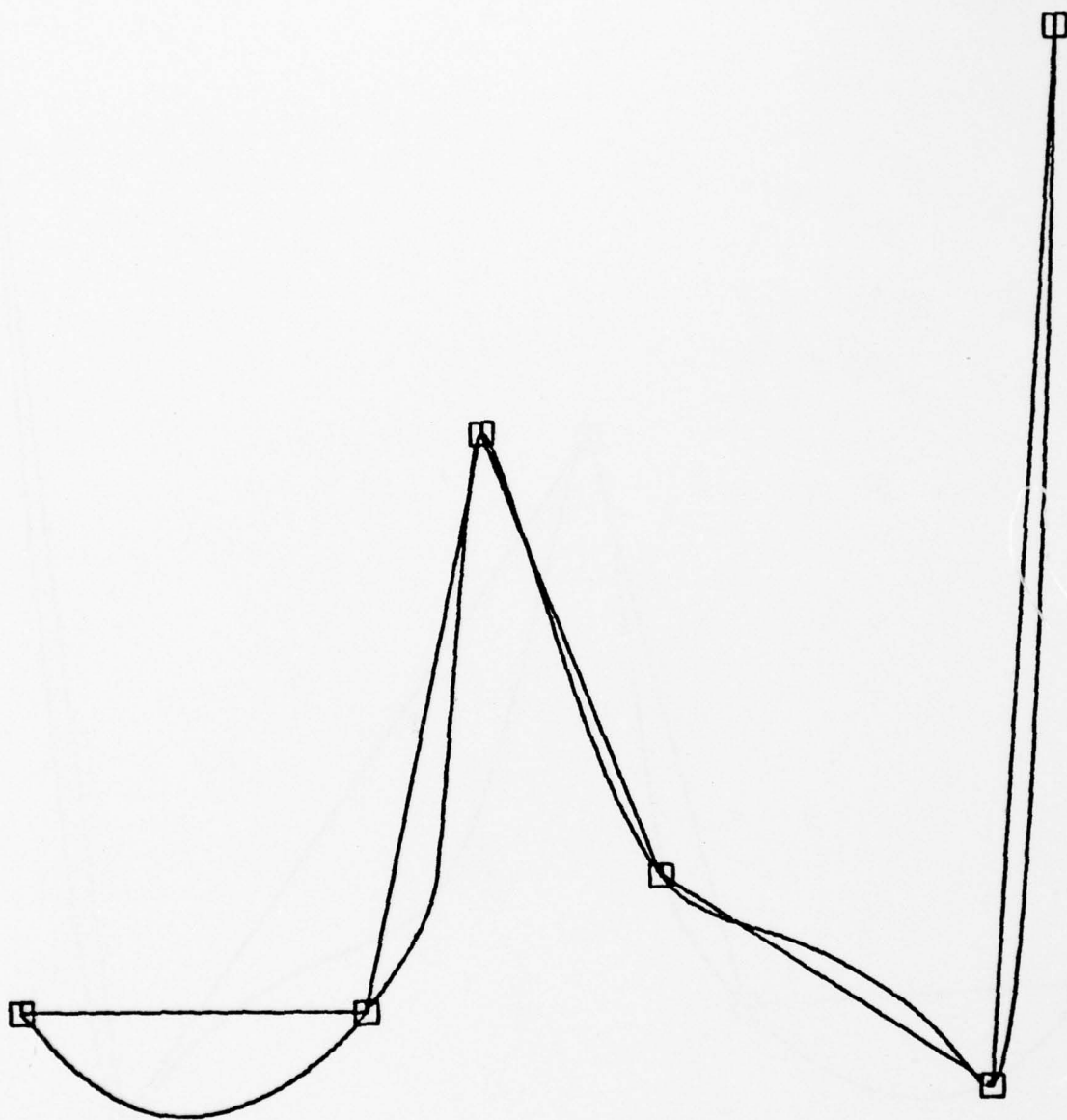
We may obtain our set of knots by selecting a subset of the data abscissas. We begin our set with the first and last abscissas, defining the first subinterval. Next, we examine the area of the triangle formed by the smoothed y values at the interval endpoints and each point within the interval, taking the abscissa corresponding to the largest triangle as the next knot, defining two more subintervals (and eliminating one). Each subinterval therefore has associated with it, a weight equal to this largest inscribed triangular area. Continuing to split subintervals with the largest weight, we may quickly pick out the major peaks and valleys in the data. We may stop adding knots after a few consecutive additions fail to violate local monotonicity, because knots near the major peaks and valleys will then have been established. The following diagrams should make this process clear.



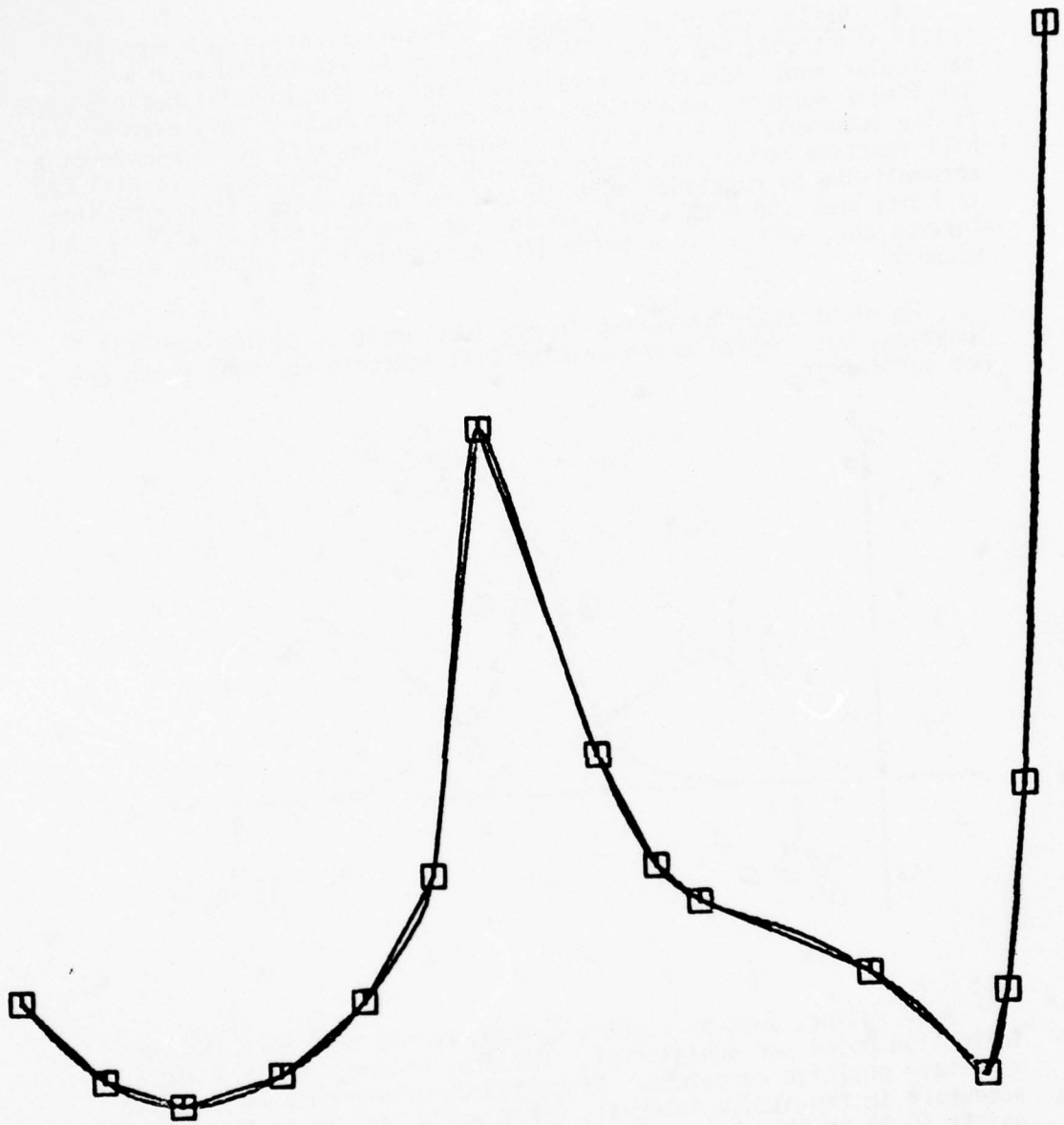






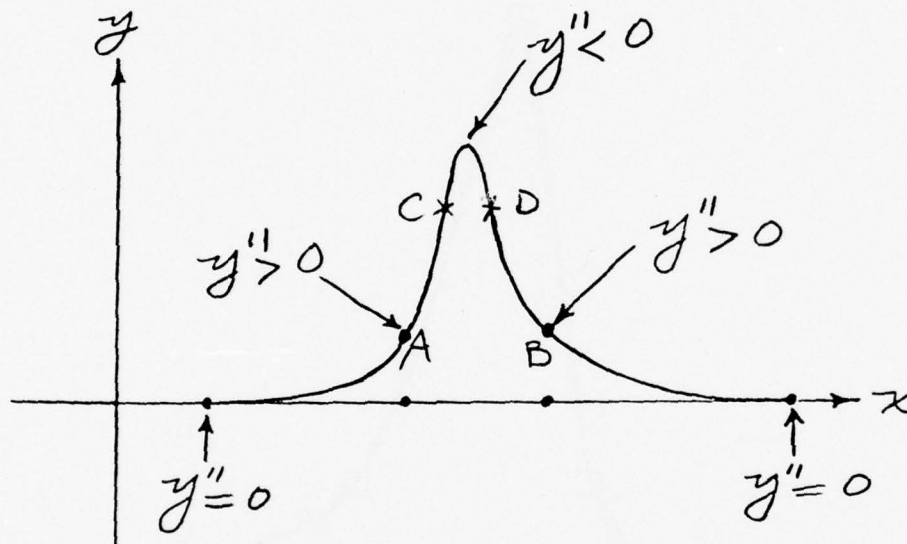


f_1



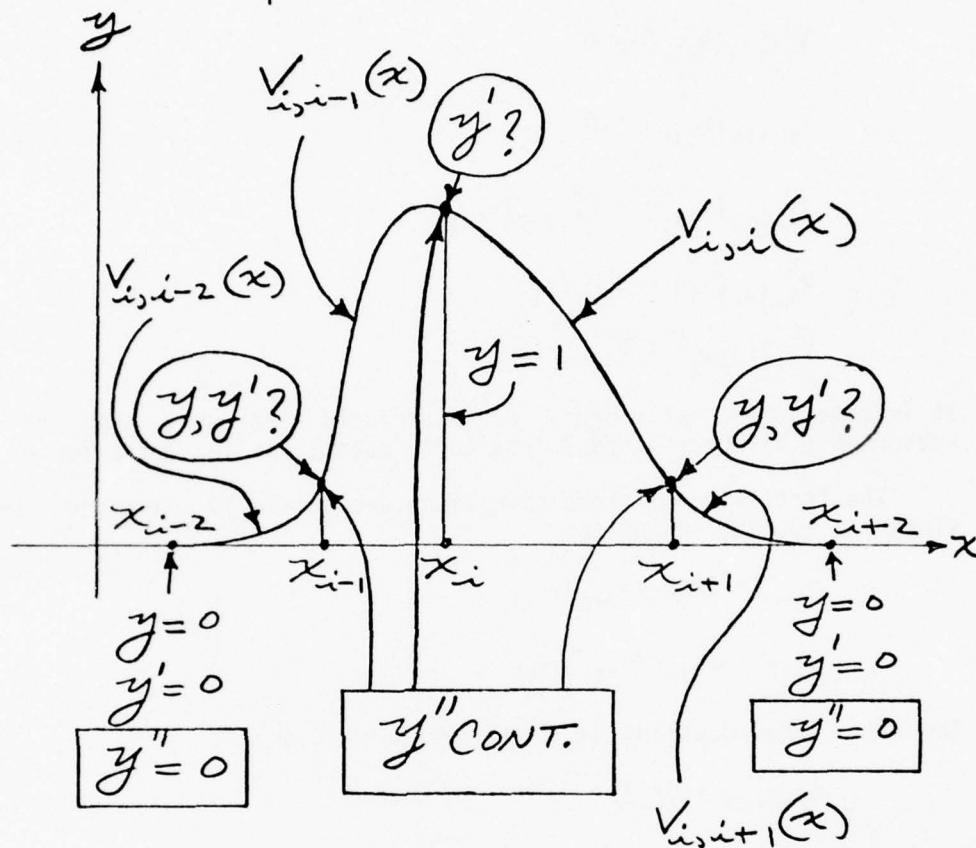
4. DEFINITION OF VPB SPLINES. VPB splines are a sequence of "hill" functions, where each hill function is associated with a particular knot (specified x value). Each of these hill functions has finite support, which means that each is non zero only over a finite interval. The objective of this section will be to construct a hill function corresponding to the i th knot. This objective will be accomplished by constructing a minimal set of y values (corresponding to knots near the i th knot), which, when interpolated by a VP spline interpolant, will yield a twice differentiable hill function of finite support.

We might start by trying to use four knots to define our hill function, but the following drawing will indicate why four knots are not sufficient.



A VP spline, just as a cubic spline, cannot have more than one inflection point per subinterval. For this reason, points A and B must have positive curvature. Since the curvature must be negative somewhere in the middle interval, there would have to be two inflection points (C,D) in this interval. This contradiction shows that four knots are insufficient. The addition of only one more knot will be sufficient to define a hill function, because no subinterval will need to have more than one inflection point.

The following diagram will aid in visualizing the definition of the i th VPB spline $V_i(x)$.



Since $V_i(x)$ consists of four different adjacent functions, $V_{i,j}(x)$ will refer to the function on the j th subinterval in particular. The circles in the drawing indicate the unknown quantities to be constructed and the boxes show the conditions with which these quantities will be determined. The i th VPB spline may therefore be defined for $3 \leq i \leq N - 2$ by:

$$V_{i,i-2}(x_{i-2}) = 0 = y_{i-2}$$

$$V_{i,i}(x_i) = 1 = y_i$$

$$V_{i,i+1}(x_{i+2}) = 0 = y_{i+2}$$

$$V'_{i,i-2}(x_{i-2}) = 0 = y'_{i-2}$$

$$V'_{i,i+1}(x_{i+2}) = 0 = y'_{i+2}$$

$$V''_{i,i-2}(x_{i-2}) = 0$$

$$V''_{i,i+1}(x_{i+2}) = 0$$

$$V''_{i,i-2}(x_{i-1}) = V''_{i,i-1}(x_{i-1})$$

$$V''_{i,i-1}(x_i) = V''_{i,i}(x_i)$$

$$V''_{i,i}(x_{i+1}) = V''_{i,i+1}(x_{i+1})$$

It is understood that y and y' are associated with $V_i(x)$ only; the subscript i has been dropped from these quantities for convenience.

The first five of these conditions are trivially satisfied; the sixth and seventh reduce to:

$$y_{i-1} = (\ell_{i-2}/m_{i-2})y'_{i-1}$$

$$y_{i+1} = -(\ell_{i+1}/n_{i+1})y'_{i+1}$$

The last three equations may therefore be written as:

$$Q_{11}y'_{i-1} + Q_{12}y'_i = Q_{14}$$

$$Q_{21}y'_{i-1} + Q_{22}y'_i + Q_{23}y'_{i+1} = Q_{24}$$

$$Q_{32}y'_i + Q_{33}y'_{i+1} = Q_{34}$$

where

$$Q_{11} = B_{i-1} + (E_{i-1}\ell_{i-2}/\ell_{i-1} - D_{i-1})/m_{i-2}$$

$$Q_{12} = C_{i-1}$$

$$Q_{14} = E_{i-1}/\ell_{i-1}$$

$$Q_{21} = A_i + D_i\ell_{i-2}/(m_{i-2}\ell_{i-1})$$

$$Q_{22} = B_i$$

$$Q_{23} = C_i + E_i \ell_{i+1} / (n_{i+1} \ell_i)$$

$$Q_{24} = D_i / \ell_{i-1} - E_i / \ell_i$$

$$Q_{32} = A_{i+1}$$

$$Q_{33} = B_{i+1} + (D_{i+1} \ell_{i+1} / \ell_i - E_{i+1}) / n_{i+1}$$

$$Q_{34} = -D_{i+1} / \ell_i$$

and the A's, B's, C's and D's are defined as in section 2.

It is not immediately obvious that the determinant of this system cannot be zero. An outline of a proof of this fact will therefore be given here. First, the B's, D's and E's may be expressed in terms of A's and C's. The Q's of the determinant will therefore be of the form:

$$Q = aA + bC$$

where a and b are not both zero. It is easily shown that if either a or b is not zero, then it must be positive, provided all m's and n's are > 2 . We have, therefore, that all Q's of the determinant are negative, since the A's and C's are all negative. The determinant of the system is given by:

$$\Delta = Q_{11}Q_{22}Q_{33} - Q_{11}Q_{23}Q_{32} - Q_{12}Q_{21}Q_{33}$$

or

$$\Delta / (Q_{11}Q_{22}Q_{33}) = 1 - (Q_{11}Q_{23}Q_{32} + Q_{12}Q_{21}Q_{33}) / (Q_{11}Q_{22}Q_{33})$$

In order to show that Δ is negative, it is sufficient to show that

$$q = (Q_{11}Q_{23}Q_{32} + Q_{12}Q_{21}Q_{33}) / (Q_{11}Q_{22}Q_{33}) < 1$$

In the process of computing the products in the numerator and denominator of q, it will be found that all terms of the numerator will be of the form aACA or bCAC, where a and b are > 0 . The denominator will also contain an AAA term and a CCC term. It is important to note that all terms of both the numerator and the denominator are negative. Now, for every term in the numerator, there is a corresponding term in the denominator whose coefficient is strictly greater than the numerator coefficient. This fact and the fact that all coefficients are positive implies that this quotient is always less than one and therefore that the determinant is always negative.

For one coefficient pair, we have the following example: The coefficient of $A_{i+1}C_iA_{i-1}$ in the numerator of q is:

$$(1+m_i\ell_{i+1}/(n_{i+1}\ell_i))(n_{i-2}-1-n_{i-2}/m_{i-2})$$

The coefficient of $A_{i+1}C_iA_{i-1}$ in the denominator of q is:

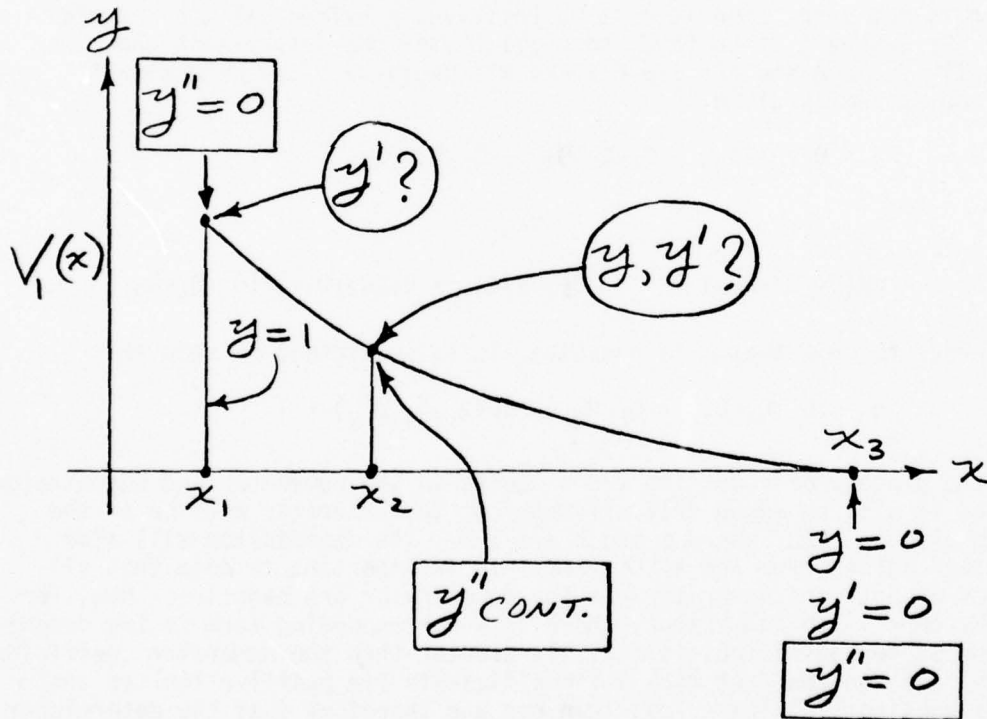
$$(n_i-1+n_i\ell_{i+1}/(n_{i+1}\ell_i))(m_i-1)(n_{i-2}-1-n_{i-2}/m_{i-2})$$

The quotient of the numerator and the denominator coefficients reduces to:

$$(n_{i+1}\ell_i+m_i\ell_{i+1})/(n_{i+1}\ell_i(m_i-1)(n_i-1)+n_i(m_i-1)\ell_{i+1})$$

In this fraction, the left and right terms of the numerator are less than the left and right terms of the denominator, respectively, since $m_i + n_i - m_in_i < 0$.

The first and second VPB splines may be defined through the following diagrams and conditions.



Conditions defining $V_1(x)$:

$$V_{1,1}(x_1) = 1 = y_1$$

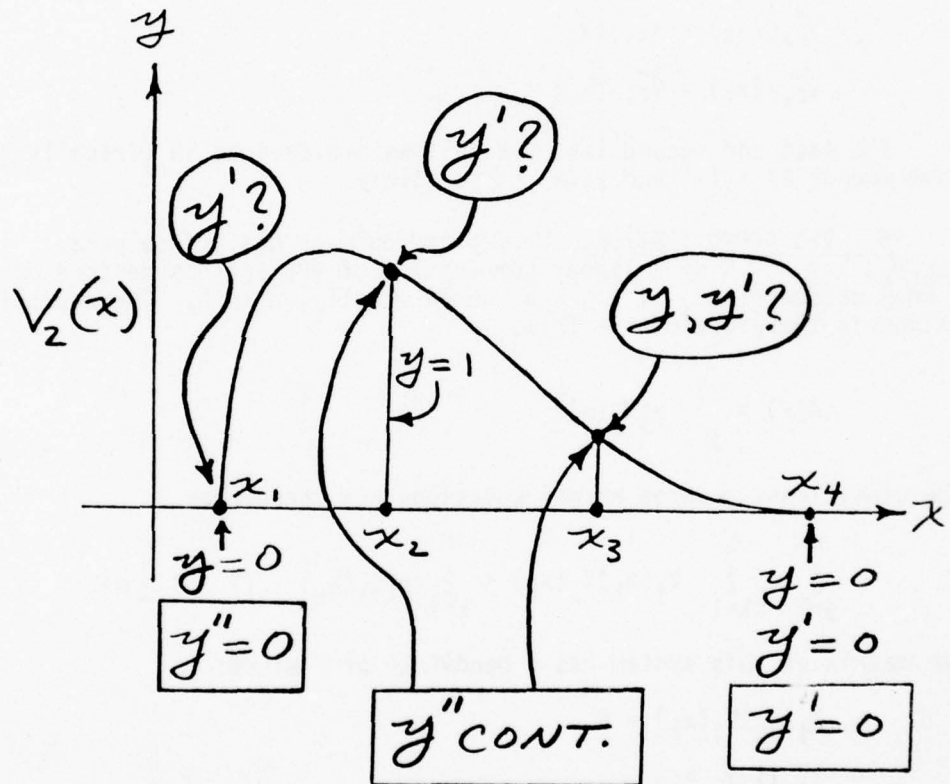
$$V_{1,2}(x_3) = 0 = y_3$$

$$V'_{1,2}(x_3) = 0 = y'_3$$

$$V''_{1,1}(x_1) = 0$$

$$V''_{1,2}(x_3) = 0$$

$$V''_{1,1}(x_2) = V''_{1,2}(x_2)$$



The conditions defining $V_2(x)$ are:

$$V_{2,1}(x_1) = 0 = y_1$$

$$V_{2,2}(x_2) = 1 = y_2$$

$$V_{2,3}(x_4) = 0 = y_4$$

$$V'_{2,3}(x_4) = 0 = y'_4$$

$$V''_{2,1}(x_1) = 0$$

$$V''_{2,3}(x_4) = 0$$

$$V''_{2,1}(x_2) = V''_{2,2}(x_2)$$

$$V''_{2,2}(x_3) = V''_{2,3}(x_3)$$

The last and second last VPB splines are defined in virtually the same manner as $V_1(x)$ and $V_2(x)$ respectively.

5. VPB APPROXIMATION. We may approximate univariate data: (x_k, y_k) $1 \leq k \leq N$ by a linear combination of VPB splines defined over a knot sequence: u_j $1 \leq j \leq n$, where usually, $n \ll N$. The approximation is therefore of the form:

$$A(x) = \sum_{j=1}^n a_j V_j(x)$$

The usual least squares normal equations are therefore:

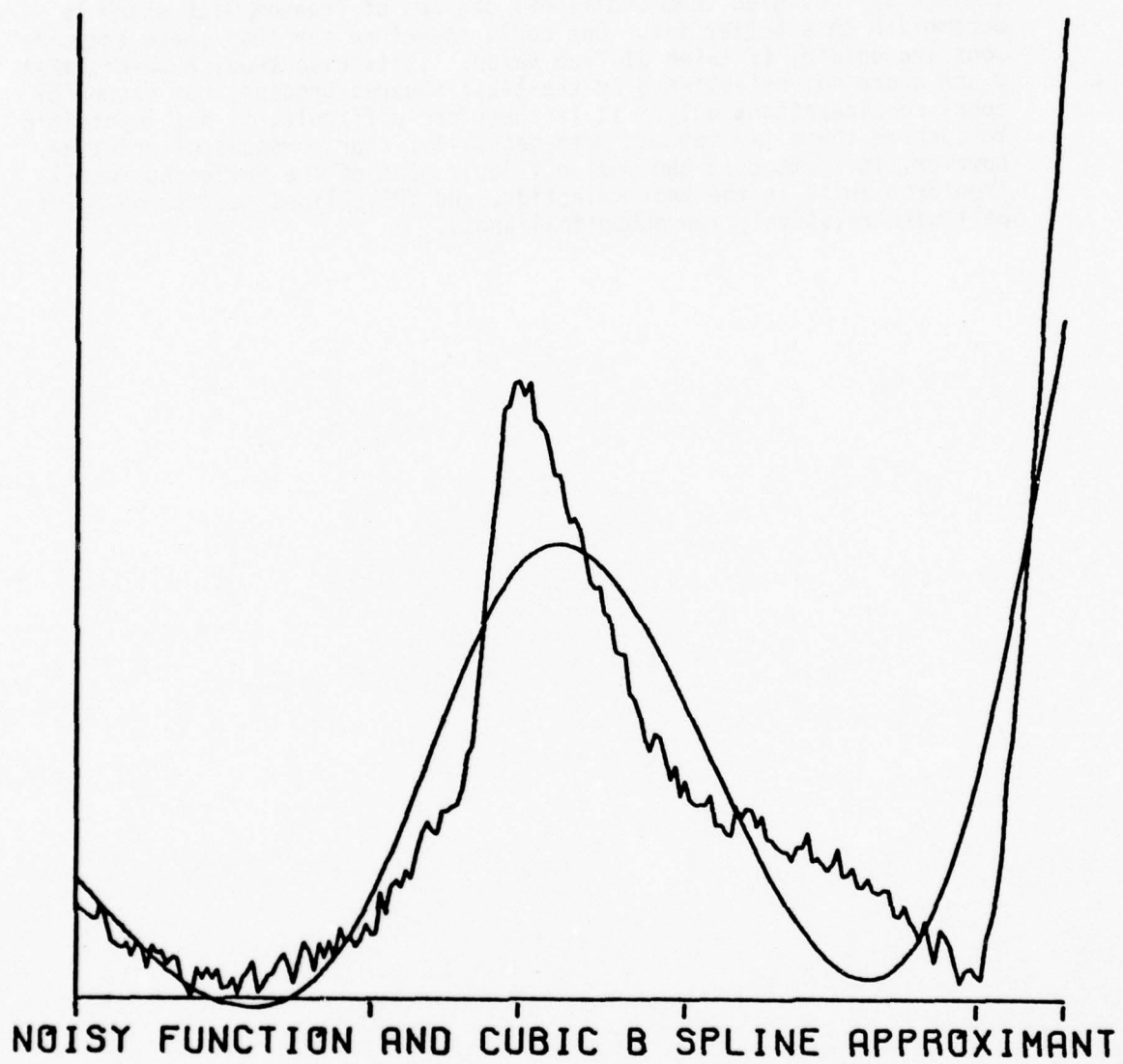
$$\sum_{j=1}^n a_j \sum_{k=1}^N V_i(x_k) V_j(x_k) = \sum_{k=1}^N y_k V_i(x_k) \quad (1 \leq i \leq n)$$

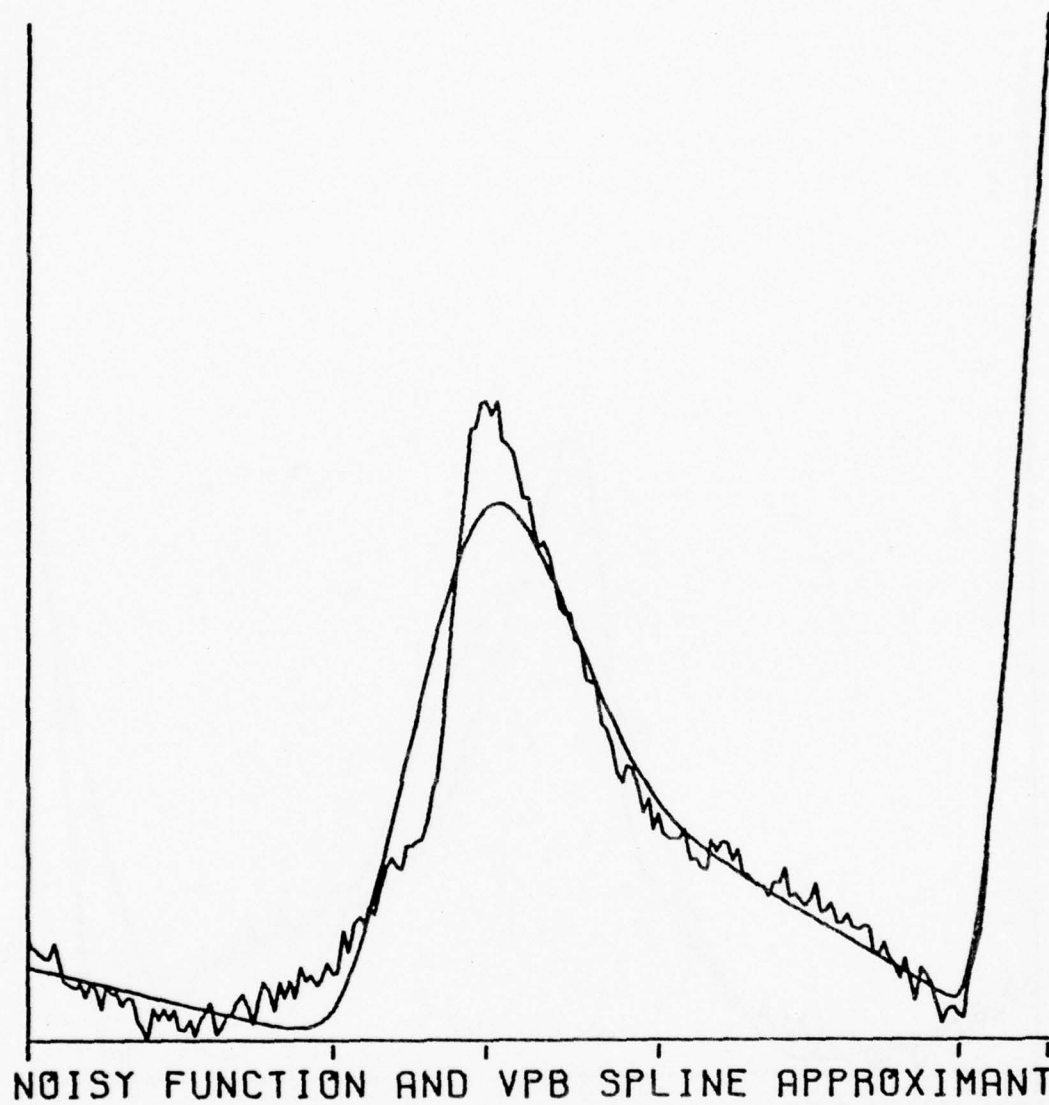
The matrix of this system has a bandwidth of 7 since:

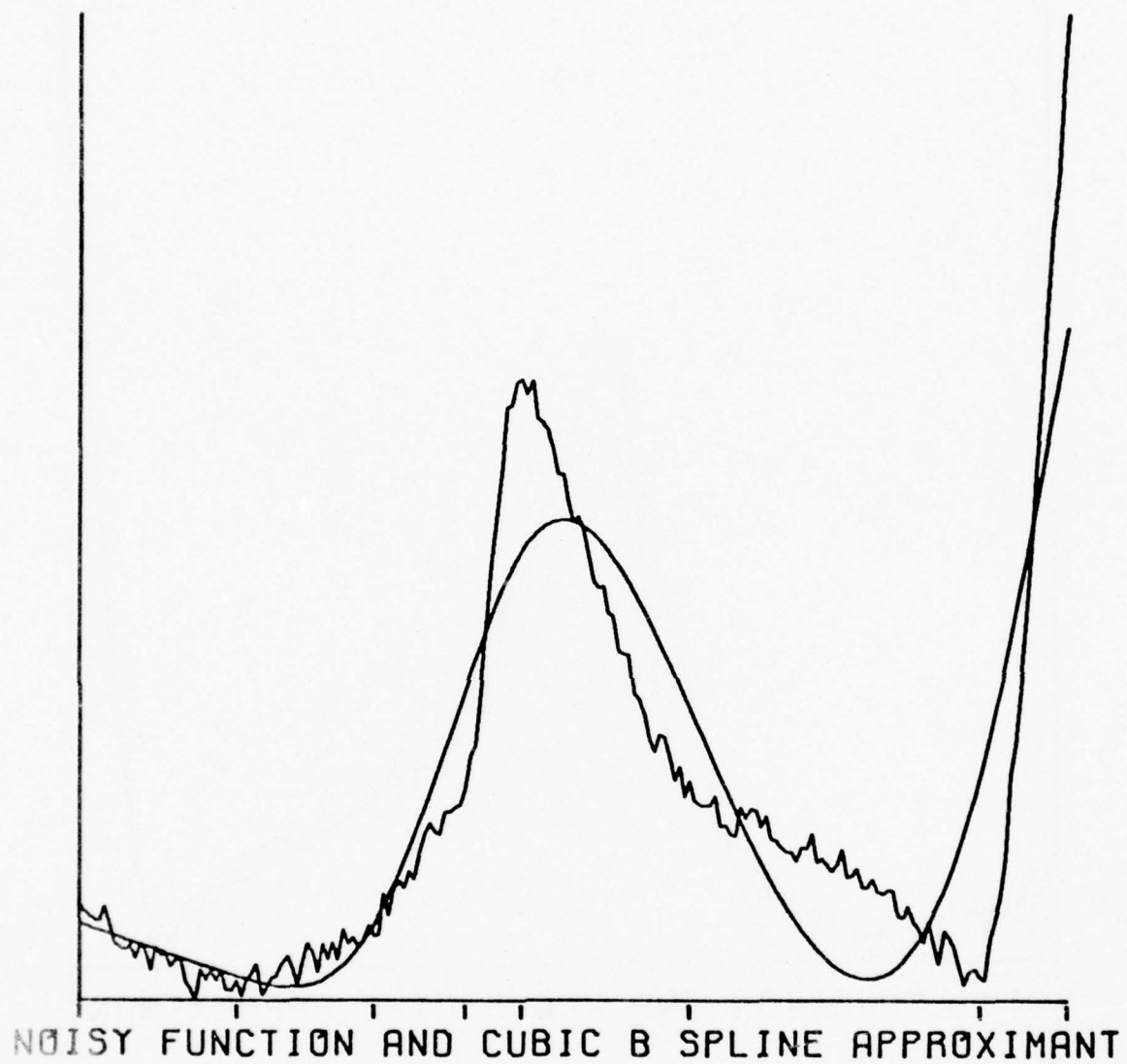
$$V_i(x_k) V_j(x_k) = 0$$

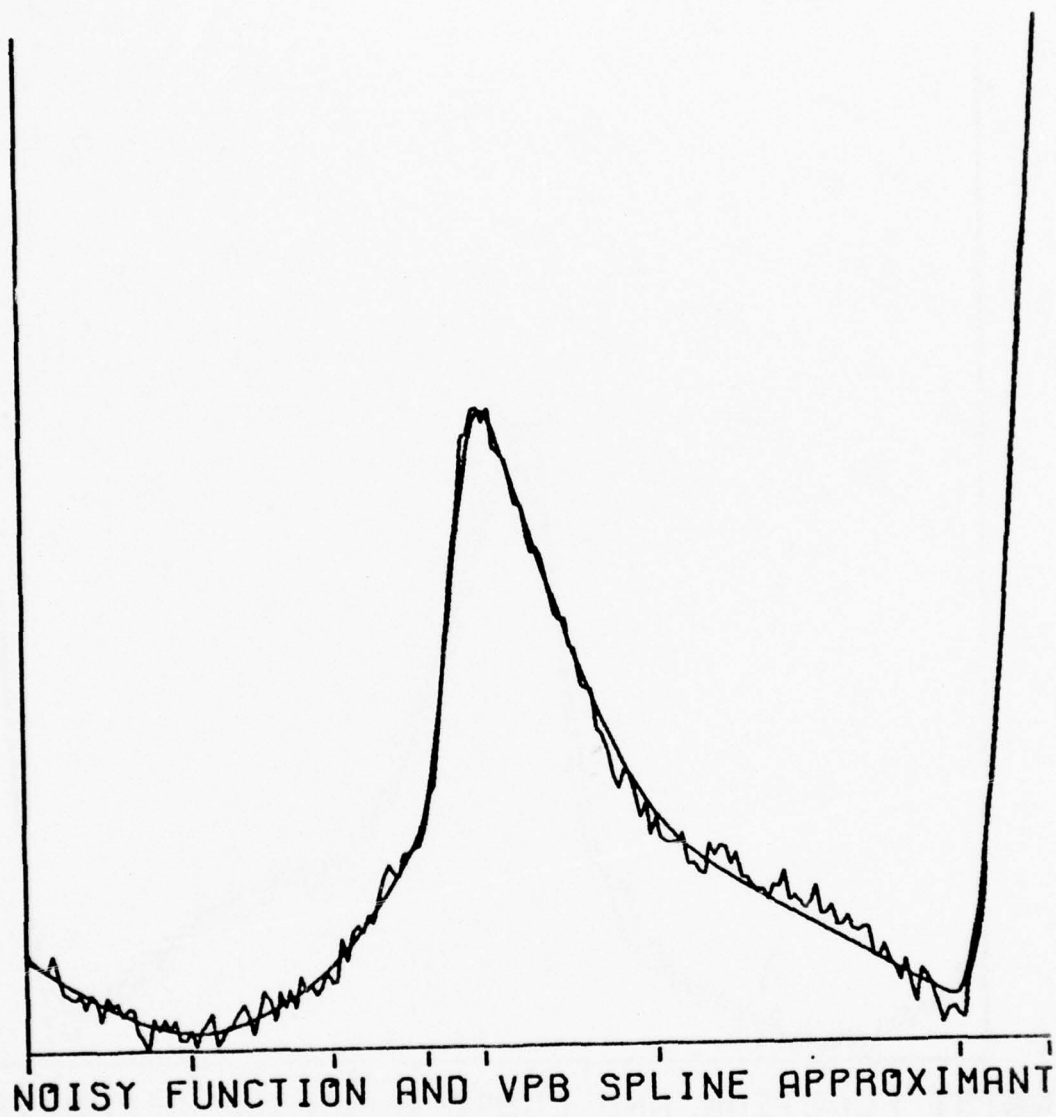
$$\text{if } |i-j| > 3$$

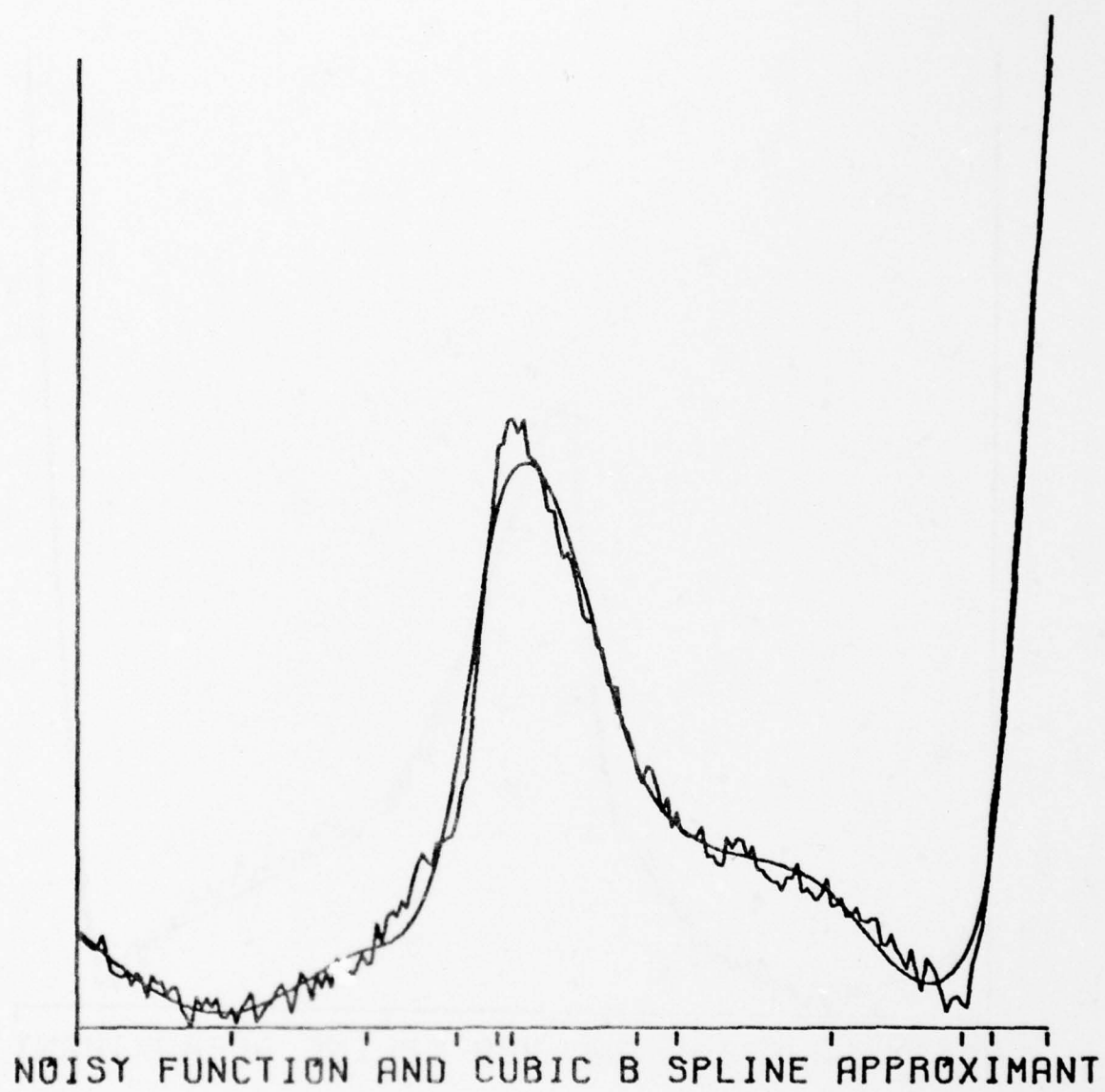
The following drawings compare the behavior of VPB and cubic B splines in fitting some noisy data. The knots used in each comparison are marked on the x axis. The VPB splines obviously give a better fit, but it should be noted that the nonlinear parameter vectors m and n of the VPB splines give them additional degrees of freedom with which to accomplish this better fit. One could therefore say that these comparisons are unfair, if taken at face value. It is also true, however, that m and n are not established by the least squares process, but rather by local considerations only. It is therefore difficult, if not impossible, to compare these two methods mathematically. For purposes of practice, however, it seems that the most difficult part of the spline approximation problem is in the knot selection, and VPB splines seem to do quite well with relatively few suboptimal knots.

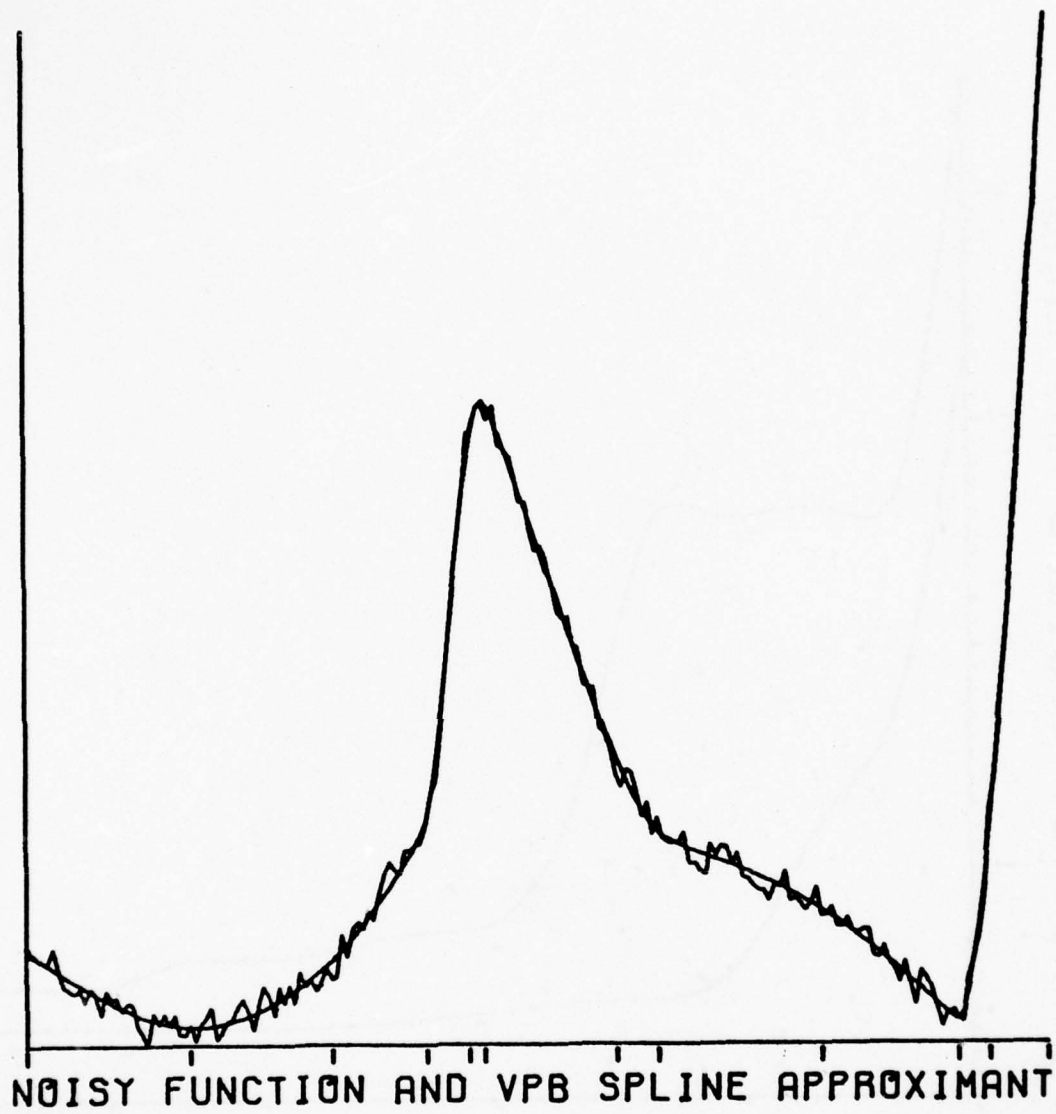


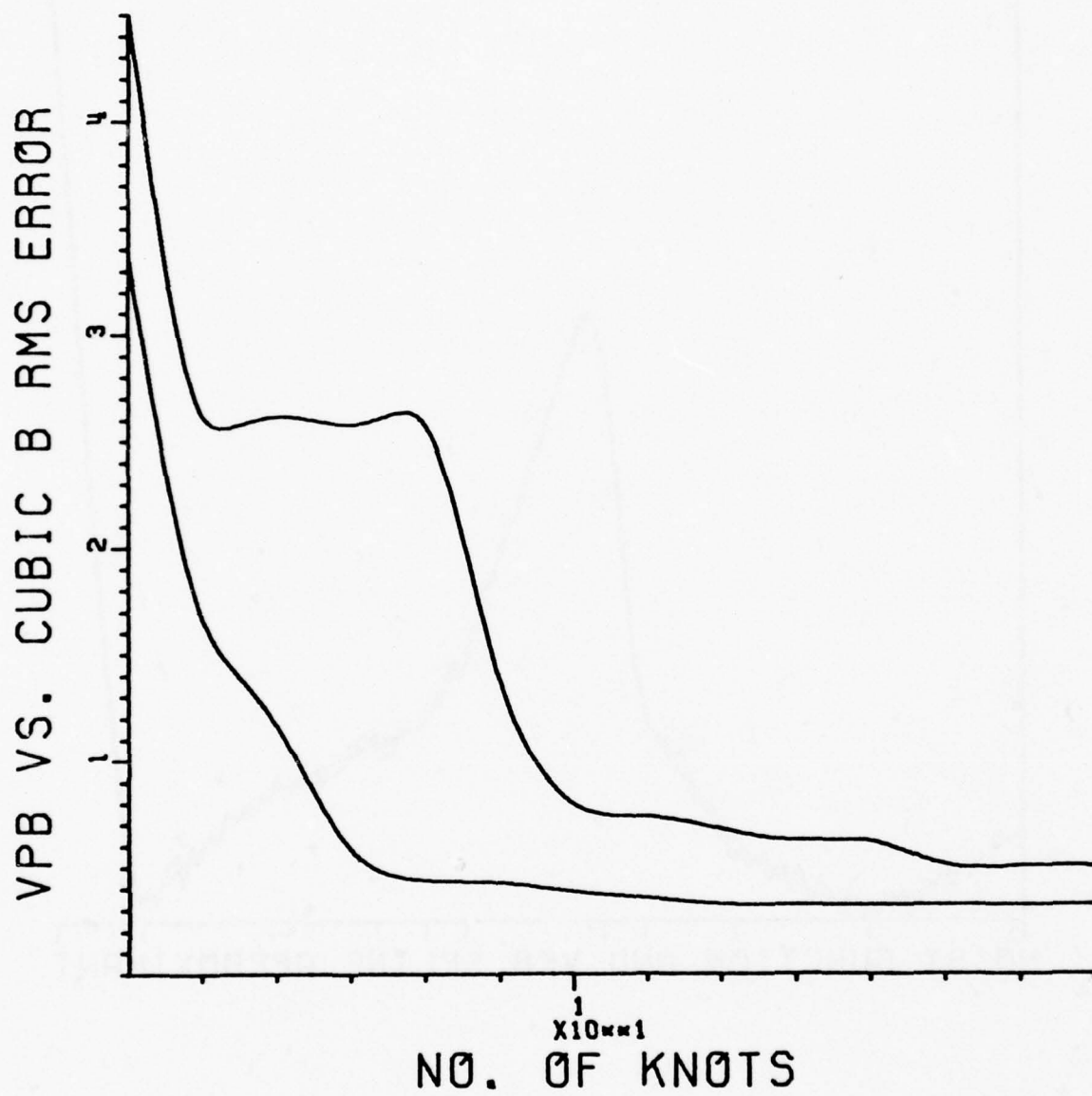












REFERENCES

1. de Boor, C., "On calculating with B-splines", J. Approx. Th., 6, pp. 50-62 (1972).
2. de Boor, C., "Package for calculating with B-splines", Univ of Wisconsin, Mathematics Research Center Technical Report #1333 (Oct 1973).
3. Chui, C., Smith, P., Ward, J., "On uniqueness of piecewise polynomial approximation", Proc. of 1975 Army Conf. of Mathematicians, pp. 141-147.
4. Prenter, P.M., Splines and Variational Methods, Wiley, New York (1975).
5. Smith, P., Hrncir, S., "Nonlinear spline regression on mini-computers", Proc. of 1976 Army Num. Anal. and Computers Conf., pp. 53-90.
6. Soanes, R., "VP-splines, an extension of twice differentiable interpolation", Proc. of 1976 Army Num. Anal. and Computers Conf., pp. 141-152.
7. _____, "Automatic numerical integration using VP-splines", Proc. of 1976 Army Conf. of Mathematicians, pp. 313-323.

BEST L_2 APPROXIMATION FROM NONLINEAR SPLINE

MANIFOLDS I - UNICITY RESULTS*

Charles K. Chui, Philip W. Smith, and Jeff Chow
Department of Mathematics
Texas A&M University
College Station, Texas 77843

ABSTRACT. Two unicity results concerning best L_2 approximation from (nonlinear) manifolds of C^0 piecewise polynomials with variable break points are given. An algorithm for obtaining the best approximants is also included.

1. **INTRODUCTION.** We will denote a knot sequence on $[0,1]$ by $\underline{t}^N = (t_0, \dots, t_{N+1})$, $0 = t_0 < t_1 < \dots < t_{N+1} = 1$. Let P^{2k} be the class of all polynomials with order $2k$ (or degree $\leq 2k-1$) and set $S_N^{2k} = \{s \in C[0,1]: \text{there is a } \underline{t}^N \text{ so that } s \in P^{2k} \text{ on } (t_i, t_{i+1}) \text{ for } i = 0, \dots, N\}$. Clearly, S_N^{2k} is a nonlinear manifold and could be represented by linear combinations of B-splines with each knot having multiplicity $2k-1$.

Although S_N^{2k} is not closed in $L_2[0,1]$ one can show using the arguments of [1,4] that if f is continuous then f has a best $L_2[0,1]$ approximant from S_N^{2k} . Thus, in this article we concentrate only on the questions of uniqueness and eventual uniqueness. In particular, in section 2 we will sketch the proofs of the following theorems.

Theorem 1. Let $f \in C^{2k}[0,1]$ with $f^{(2k)} > 0$ on $[0,1]$. Suppose that $f^{(2k)}$ is logarithmic concave on $(0,1)$. Then for every positive integer N , f has a unique best $L_2[0,1]$ approximant from S_N^{2k} .

The logarithmic concavity condition is not necessary when we have "many" knots.

Theorem 2. Let $f \in C^{(2k+3)}[0,1]$ with $f^{(2k)} > 0$ on $[0,1]$. Then there exists an N_0 such that for each $N > N_0$, f has a unique best $L_2[0,1]$ approximant from S_N^{2k} .

It should be emphasized that N_0 depends on f . This is the idea of eventual uniqueness. We will discuss in section 3 a method of computing the best $L_2[0,1]$ approximants and in section 4 some numerical examples

*This research was supported by the U.S. Army Research Office under Grant No. DAHC 04-75-G-0186.

will be presented.

2. SKETCH OF PROOFS. In this section we will sketch the proofs of Theorems 1 and 2. As will be evident the main tool is topological degree theory.

Let $\Sigma^N \subset \mathbb{R}^N$ be the open simplex $\{\underline{t}^N = (t_1, \dots, t_N) : 0 < t_1 < \dots < t_N < 1\}$. The \underline{t}^N will denote the interior knot sequence of the various splines. We are interested in studying the uniqueness of an $s \in S_N^{2k}$ so that

$$\inf \left\{ \int_0^1 |g-f|^2 : g \in S_N^{2k} \right\} = \int_0^1 |s-f|^2.$$

Such an s will be called a best $L_2[0,1]$ approximant to f from S_N^{2k} .

The following two lemmas will be useful in formulating the proofs of Theorems 1 and 2.

Lemma 1. Let $f \in C^{(2k)}[0,1]$, $f^{(2k)} > 0$, and $s \in S_N^{2k}$ be a best $L_2[0,1]$ approximant to f from S_N^{2k} . Let s have interior knots at $\{t_i\}_{i=1}^N$. Then the restriction of s to any of the intervals (t_i, t_{i+1}) , $i=0, \dots, N$, is a best L_2 approximation to f on the interval (t_i, t_{i+1}) from P^{2k} . That is

$$\int_{t_i}^{t_{i+1}} [(f-s)(t)](t^j) dt = 0, \quad j=0, 1, \dots, 2k-1; \quad i=0, \dots, N.$$

This lemma is easily established by differentiating the error with respect to the knots. A minor but nagging point in the proof of this lemma requires one to show that if s is a best $L_2[0,1]$ approximant to f from S_N^{2k} then $s^{(2k-1)}(t_i^+) \neq s^{(2k-1)}(t_i^-)$. This result can be obtained via the zero counting techniques found in [5].

The second lemma tells us that all the knots of the best approximation are active.

Lemma 2. Let f be a continuous function on $[0,1]$ with $f \notin S_N^{2k}$ and let $s \in S_N^{2k}$ be a best $L_2[0,1]$ approximation to f from S_N^{2k} . Then $s \notin S_{N-1}^{2k}$.

Using Lemmas 1 and 2 we can formulate a necessary condition for the knots, \underline{t}^N , to be the knots of a best $L_2[0,1]$ approximant to f from S_N^{2k} provided $f^{(2k)} > 0$. In particular, let ℓ_i be the best $L_2[0,1]$ approximant to f on the interval (t_i, t_{i+1}) from P^{2k} , $i=0, \dots, N$. Then a necessary condition for the t_i 's to be the knots of a best $L_2[0,1]$ approximant to f

is that $\ell_i(t_{i+1}) = \ell_{i+1}(t_{i+1})$ for $i=0, \dots, N-1$. That is, we would like to show that the map

$$\underline{F} : \Sigma^N \rightarrow \mathbb{R}^N$$

defined by

$$\underline{F}(\underline{t}^N) = -(\ell_1(t_1) - \ell_0(t_1), \dots, \ell_N(t_N) - \ell_{N-1}(t_N))$$

has precisely one solution.

Letting $\Delta t_i = t_{i+1} - t_i$, we see that

$$\begin{aligned} \underline{F}_1(\underline{t}^N) = & -(\Delta t_1)^{2k}/(2k-1)! \int_0^1 \tau^{2k-1}(1-\tau)^{2k} f^{(2k)}(\tau \Delta t_1 + t_1) \\ & + (\Delta t_{i-1})^{2k}/(2k-1)! \int_0^1 \tau^{2k}(1-\tau)^{2k-1} f^{(2k)}(\tau \Delta t_{i-1} + t_{i-1}) d\tau. \end{aligned}$$

Using the arguments in [1] one can now see that if $f^{(2k)} > 0$ and $\log f^{(2k)}$ is concave then the degree of F with respect to Σ^N and the point $(0, \dots, 0)$ is 1 and that this counts the number of solutions. Similarly, Theorem 2 is proved by showing that the degree is 1 and that eventually (i.e. for large N) this is the number of solutions. Details and related results can be found in [3].

3. NUMERICAL ALGORITHM. Let $\underline{t}^N \in \Sigma^N$ and k be a positive even integer. For $i=1, \dots, N$, set

$$\begin{aligned} F_i = F_i(\underline{t}^N, f) = & (\Delta t_{i-1})^k/(k-1)! \int_0^1 \tau^{k-1}(1-\tau)^{k-1} f^{(k)}(\tau \Delta t_{i-1} + t_{i-1}) d\tau \\ & - (\Delta t_i)^k/(k-1)! \int_0^1 \tau^{k-1}(1-\tau)^k f^{(k)}(\tau \Delta t_i + t_i) d\tau \end{aligned}$$

where $\Delta t_j = t_{j+1} - t_j$. The Jacobian matrix $[\partial F_i / \partial t_j]$ of (F_1, \dots, F_N) is tridiagonal and its nonzero entries can be written as follows:

$$\partial F_i / \partial t_{i-1} = -k(\Delta t_{i-1})^{k-1}/(k-1)! \int_0^1 \tau^{k-1}(1-\tau)^k f^{(k)}(\tau \Delta t_{i-1} + t_{i-1}) d\tau, \quad 2 \leq i \leq N$$

$$\begin{aligned} \partial F_i / \partial t_i = & (\Delta t_{i-1})^{k-1}/(k-1)! \int_0^1 \tau^k(1-\tau)^{k-2} (k\tau-1) f^{(k)}(\tau \Delta t_{i-1} + t_{i-1}) d\tau \\ & + (\Delta t_i)^{k-1}/(k-1)! \int_0^1 \tau^{k-2}(1-\tau)^k (k(1-\tau)-1) f^{(k)}(\tau \Delta t_i + t_i) d\tau, \end{aligned}$$

$$1 \leq i \leq N,$$

$$\partial F_i / \partial t_{i+1} = -k(\Delta t_i)^{k-1} / (k-1)! \int_0^1 \tau^{k(1-\tau)^{k-1}} f^{(k)}(\tau \Delta t_i + t_i) d\tau, \quad 1 \leq i \leq N-1.$$

Once k , \underline{t}^N and $f^{(k)}$ are known, the numerical iterations by Newton's method can be easily calculated to get a solution of $F_i = 0$, $i=1, \dots, N$. At each iteration, we use a Gaussian quadrature formula to approximate the integrals which appear in the Jacobian matrix $[\partial F_i / \partial t_j]$ and in (F_1, \dots, F_N) .

In the following we will describe a numerical algorithm to locate a zero of (F_1, \dots, F_N) .

- Step 1. Initialize \underline{t}^N .
- Step 2. Check \underline{t}^N to be sure it is in sequence.
- Step 3. Check the number of iterations and the size of relative error between two consecutive iterations.
- Step 4. Build (F_1, \dots, F_N) and the tridiagonals of $[\partial F_i / \partial t_j]$ via Gaussian quadrature formula.
- Step 5. Solve $Ax = b$ where $[a_{ij}] = [\partial F_i / \partial t_j]$ and $b_i = F_i$.
- Step 6. Set $t_i = t_i - x_i$, where x_i is obtained from Step 5. Then go to Step 2.

Remarks. For lack of information in Step 1, one usually starts with equally spaced \underline{t}^N . In order to compare with Burchard and Hale's asymptotic constant [2], as will be done in the next section, we use his balanced mesh as initial knots. Step 2 seems redundant. However, it serves as a safeguard against any undesired mesh.

4. NUMERICAL EXAMPLES. Since the approximation is taken in the L^2 -norm, for given $f^{(k)}$, $k=2$, one can easily calculate Burchard and Hale's asymptotic constant

$$B_{2,k}(f) = C_{2,k} \|f''\|_{\sigma} = \frac{1}{12\sqrt{5}} \|f''\|_{\sigma}$$

where $\sigma = (k+2^{-1})^{-1} = 2/5$ and $\|f''\|_{\sigma} = (\int |f''|^{\sigma})^{1/\sigma}$. A similar numerical scheme as above will work for $k=4, 6, \dots$

In both examples below, we are going to consider only splines of order 2, that is, from S_N^2 . Let $f \in C^2[0,1]$. For $N=1, \dots, 47$, we try to calculate

$$(N+1)^2 \inf_{s \in S_N^2} \|f-s\|_2$$

and note that it "converges" to $B_{2,2}(f)$ as N increases.

A mesh $\underline{t}^N \in \Sigma^N$ is called balanced with respect to f if

$$\int_{t_{i-1}}^{t_i} |f^{(k)}|^{\sigma} = (N+1)^{-1} \int_0^1 |f^{(k)}|^{\sigma}, \quad i=1, \dots, N+1.$$

where $\sigma = (k+2^{-1})^{-1}$ [2]. We take such \underline{t}^N as our initial guess for the above algorithm. Let $\tau = \max_{1 \leq i \leq N+1} (t_i - t_{i-1})$ and let x_i be the increment in t_i by Newton's method. Further iterations need not be carried out if

$$\max_{1 \leq i \leq N} |x_i| \leq 10^{-10} \tau.$$

Actually, it takes less than 6 iterations in all cases to satisfy the above terminating criterion.

At the end we will compare $(N+1)^2 \|f-s\|_2^2$ among three different piecewise polynomials S_j , namely, (i) $S_1 = S_1(\underline{t}^N) \in S_N^2$ such that $\|f-S_1\|_2 \leq \|f-g\|_2$ for all $g \in S_N^2$, (ii) $S_2 = S_2(\underline{t}^N) \in S_N^2$ such that \underline{t}^N is balanced, and (iii) $S_3 = S_3(\underline{t}^N) \notin S_N^2$ such that \underline{t}^N is balanced and S_3 is allowed to be discontinuous at each of the knots. In each case best linear approximants are computed.

We consider the following two examples:

- (a) $f(x) = x^3/6$ and $f''(x) = x$ on $[0,1]$;
- (b) $f(x) = -4x^{1/2}$ and $f''(x) = x^{-3/2}$ on $(0,1]$.

Note that in example (a) $\|f-s\|_2$ and the integrals which form the entries of (F_1, \dots, F_N) and $[\partial F_i / \partial t_j]$ can be accurately computed by using the Gaussian quadrature formula with 4 Gaussian points. The reason is simply that the integrand in each integral is a polynomial of degree ≤ 6 . However, in example (b) f'' is not only not a polynomial but also unbounded on $(0,1]$. Fortunately, the integral involved with f'' over (t_0, t_1) is under control, because f'' has to be multiplied by a positive weight function with a double zero at t_0 . To gain more accurate approximation of each integral in example (b), we subdivide the interval in question into four subintervals and then apply the Gaussian quadrature formula on each subinterval.

Before tabulating our results, we calculate Burchard and Hale's

asymptotic constant for each example as follows:

$$[B_{2,2}(x^3/6)]^{5/2} = \frac{1}{12\sqrt{5}} (5/7)^{5/2} \doteq .016069918,$$

$$[B_{2,2}(-4x^{1/2})]^{5/2} = \frac{1}{12\sqrt{5}} (5/2)^{5/2} \doteq .368284782.$$

Table 1 below corresponds to example (a) and Table 2 is for example (b). In both tables, C_1 , C_2 and C_3 represent three constants $(N+1)^2 \|f-s\|_2$ among three different piecewise polynomials mentioned above.

Table 1

<u>N+1</u>	<u>C₁</u>	<u>C₂</u>	<u>C₃</u>
2	.01746675	.01765997	.01759154
4	.01676272	.01689968	.01686163
6	.01653038	.01663083	.01660505
8	.01641471	.01649357	.01647049
10	.01634548	.01641029	.01639462
12	.01629940	.01635437	.01634127
14	.01626652	.01631423	.01630298
16	.01624188	.01628402	.01627416
18	.01622273	.01626046	.01625168
20	.01620742	.01624157	.01623366
22	.01619490	.01622609	.01621889
24	.01618446	.01621317	.01620656
26	.01617564	.01620222	.01619612
28	.01616807	.01619283	.01618716
30	.01616152	.01618469	.01617939
32	.01615579	.01617755	.01617259
34	.01615073	.01617125	.01616658
36	.01614624	.01616565	.01616123
38	.01614222	.01616063	.01615645
40	.01613860	.01615612	.01615214
42	.01613532	.01615203	.01614824
44	.01613235	.01614831	.01614470
46	.01612963	.01614492	.01614146
48	.01612714	.01614180	.01613849

Table 2

<u>N+1</u>	<u>C₁</u>	<u>C₂</u>	<u>C₃</u>
2	.24631921	.28220634	.27461178
4	.29629390	.32423129	.32094308
6	.31722593	.33869744	.33660287
8	.32872871	.34601495	.34447856
10	.33600192	.35043162	.34921859
12	.34101598	.35338673	.35238464
14	.34468195	.35550266	.35464902
16	.34747905	.35709237	.35634888
18	.34968345	.35833044	.35767192
20	.35146549	.35932191	.35873094
22	.35293594	.36013378	.35959779
24	.35416996	.36081080	.36032042
26	.35522031	.36138398	.36093207
28	.35612517	.36187551	.36145647
30	.35691280	.36230168	.36191105
32	.35760461	.36267471	.36230889
34	.35821707	.36300396	.36265997
36	.35876310	.36329670	.36297210
38	.35925295	.36355870	.36325141
40	.35969486	.36379454	.36350281
42	.36009554	.36400797	.36373030
44	.36046051	.36420203	.36393712
46	.36079434	.36437924	.36412598
48	.36110085	.36454171	.36429911

References

1. D. L. Barrow, C. K. Chui, P. W. Smith, J. D. Ward, Unicity of Best Mean Approximation by Second Order Splines with Variable Knots, in manuscript and announced in Bull. Amer. Math. Soc. as "Unicity of best L_2 approximation by second-order splines with variable knots", Vol. 83,² (1977) to appear.
2. H. G. Burchard and D. F. Hale, Piecewise Polynomial Approximation on Optimal Meshes, Journal of Approximation Theory, Vol. 14, No. 2, June 1975, 128-147.
3. J. Chow, Ph. D. Thesis, Texas A&M University, to appear.
4. C. K. Chui, P. W. Smith, J. D. Ward, On the smoothness of best L_2 approximants from nonlinear spline manifolds, Math. Comp. 31 (1977), 17-23.
5. S. Karlin, C. A. Micchelli, A. Pinkus, I. J. Schoenberg, Studies in Spline Functions and Approximation Theory. Academic Press, New York, 1976.

BEST L_2 APPROXIMATION FROM NONLINEAR
SPLINE MANIFOLDS II - APPLICATION TO
OPTIMAL QUADRATURE FORMULA*

Charles K. Chui, Philip W. Smith and Joseph D. Ward
Department of Mathematics
Texas A&M University
College Station, Texas 77843

ABSTRACT. It is shown that for certain positive weight functions and certain boundary conditions the corresponding optimal quadrature formula is unique.

1. **INTRODUCTION.** In this paper we use the notation as well as the as the results in [1] to study the unicity and existence of certain optimal quadrature formulae. The results and proofs will be presented in the next section.

2. **OPTIMAL QUADRATURE FORMULAE.** Let $s = s(t^N) \in S_N^2$, $w \in C^2[0,1]$, and $e = w - s$. If $f \in C^2[0,1]$ then we can define the linear functional R by

$$\begin{aligned} R(f) &= \int_0^1 f''[w-s] \\ &\equiv \int_0^1 f''e. \end{aligned}$$

If we integrate this formula by parts twice, we obtain

$$(2.1) \quad R(f) = [f'e]_0^1 - [e'f]_0^1 - \sum_{i=1}^N A_i f(t_i) + \int_0^1 fw''.$$

where $A_i = [s']_{t_i^-}^{t_i^+}$. That is,

$$R(f) = \int_0^1 fw'' - \sum_{i=0}^1 \sum_{j=0}^1 C_{ij} f^{(j)}(i) - \sum_{i=1}^N A_i f(t_i),$$

and hence we think of R as an error functional for a quadrature formula with weight w . The interested reader should see [4] and the references therein. This quadrature formula is clearly exact for linear polynomials so that e could be thought of as the Peano kernel for the quadrature error.

Schoenberg [4] has coined the phrase optimal quadrature formula in

*This research was supported by the U. S. Army Research Office under Grant No. DAHC 04-75-G-0186.

reference to that formula which has minimal L_2 Peano kernel with a fixed number of knots and, perhaps, some boundary conditions built in. For instance, one might require $e(0) = e(1) = 0$ so that f' does not appear in the formula or one might require $e(0) = e(1) = e'(0) = e'(1)$ yielding what Schoenberg calls an "open" formula.

Notice that minimizing the L_2 norm of e over all possible $s \in S_N^2$ corresponds to minimizing

$$(2.2) \quad \sup\{|R(f)| : \int_0^1 (f'')^2 \leq 1\},$$

and hence could be thought of as choosing the best points to evaluate f in order to approximate $\int_0^1 fw''$. If w'' is positive this, of course, means that w is convex which allows us to use the results of [2] or [3]. We will sketch the proof of the following theorem

Theorem 1. Let $w'' > 0$ and $\log w''$ be concave on $[0,1]$. Then for any of the boundary conditions

- i) $e(0)$ and $e(1)$ unrestricted,
- ii) $e(0) = e(1) = 0$, or
- iii) $e(0) = e(1) = e'(0) = e'(1) = 0$,

there is a unique optimal quadrature formula of the type (2.1) minimizing (2.2).

Since minimizing (2.2) is equivalent to approximating w from S_N^2 the results of [1] or [2] immediately imply that there is a unique best approximant to w from S_N^2 and hence a unique optimal quadrature formula.

Part (i) follows directly from the results of [1]. In ii) a few results must be established since the arguments in [1] don't carry over directly. Hereafter \bar{S}_N^2 will denote $\{s \in S_N^2 : s(0) = s(1) = 0\}$. The conditions that $w'(0)$ and $w'(1)$ are finite ensure that w has a best approximant from \bar{S}_N^2 , if we assume $w(0) = w(1) = 0$. Then $w(t) \leq 0$ on $[0,1]$, and it is easily seen that the first and last linear segment of any best approximant must lie in the triangle formed by the x -axis and the lines tangent to w at $t=0$ and $t=1$. The fact that the knots are simple follows from an argument in [3].

We next show that if s^* is a best approximant to w from \bar{S}_N^2 with knot

sequence $0 = t_0 < t_1 < \dots < t_N < t_{N+1} = 1$, then the error $e = w - s^*$ is orthogonal to linear functions between t_i and t_{i+1} for $1 \leq i \leq N-1$ and on the intervals $[0, t_1]$ and $[t_N, 1]$, $w - s^*$ is orthogonal to the linear functions which vanish at $t = 0$ and $t = 1$ respectively. We establish our claims by first noting that the dimension of \bar{S}_N^2 is equal to N and that every spline in \bar{S}_N^2 is of the form

$$s(x) = \sum_{i=1}^N A_i (x - t_i)_+ - \sum_{i=1}^N A_i (1 - t_i) x.$$

Define $G(A_1, \dots, A_N, t_1, \dots, t_N) = \int_0^1 [w - s]^2 dx$ where s is any spline in \bar{S}_N^2 . Suppose s^* is the best approximant to w having the form

$$s^*(x) = \sum_{i=1}^N A_i^* (x - t_i^*)_+ - \sum_{i=1}^N A_i^* (1 - t_i^*) x.$$

By the usual variational arguments one now may easily establish the orthogonality claims. Finally it may be shown that Lemma 3 of [1] goes through under our assumptions on w . Thus there is a unique optimal quadrature formula of the type (2.1) with the additional conditions $e(0) = e(1) = 0$.

Part (iii) may now be derived using slight modifications of the arguments employed for (ii).

We wish to turn our attention to eventually unique optimal quadrature formulae. By this we mean that for fixed w and some integer N_0 , the best approximants $s_N^* \in \bar{S}_N^2$ to w are unique for $N \geq N_0$. We now may state

Theorem 2. Let $w'' > 0$ on $[0, 1]$. Then for any of the boundary conditions

- (i) $e(0)$ and $e(1)$ unrestricted,
- (ii) $e(0) = e(1) = 0$, or
- (iii) $e(0) = e(1) = e'(0) = e'(1) = 0$,

there are eventually unique optimal quadrature formula of the type (2.1) minimizing (2.2).

We sketch a proof of this theorem. As in Theorem 1, (i) follows from the results of [1] and (iii) will follow by arguments similar to those used in establishing (ii). The key to this is in showing that Proposition 1 of [1] is applicable to our situation.

Proposition 1. Let $A = (a_{ij})$ be a tridiagonal $N \times N$ real matrix with positive diagonal entries. Then if

$$(2.2) \quad a_{n,n-1} a_{n-1,n} \leq a_{n,n} a_{n-1,n-1} (1 + \pi^2/4N^2)/4$$

for $n = 2, \dots, N$ it follows that $\det A > 0$.

Hereafter we use the notation of section 3 in [1]. Our aim is to show that the determinant of $J(F(\underline{t}^N))$ is positive where $J(F(\underline{t}^N))$ refers to the Jacobian matrix of $F(\underline{t}^N)$. The only thing to be checked is that equation (2.2) is valid in the case $n = 2$ or $n = N$ for it is only here that our equations differ from those in [1]. We restrict our attention to $n = 2$. For sufficiently smooth w , a Taylor expansion shows that $w''(\Delta t_1) = K + \mathcal{O}(\Delta t_1)$ where Δt_1 goes to zero as the number of knots gets large. Thus, for the eventual uniqueness problem we may assume $w = x^2$. Hence,

$$(2.3) \quad F_1(\underline{t}^N) = (\Delta t_0)^2 \int_0^1 \frac{\tau(1-\tau)^2}{12} \cdot 2d\tau - (\Delta t_1)^2 \int_0^1 \frac{\tau(1-\tau)^2}{6} \cdot 2d\tau$$

$$(2.4) \quad F_2(\underline{t}^N) = (\Delta t_1)^2 \int_0^1 \frac{\tau(1-\tau)^2}{6} \cdot 2d\tau - (\Delta t_2)^2 \int_0^1 \frac{\tau(1-\tau)^2}{6} \cdot 2d\tau$$

Easy computations lead to

$$\frac{\partial F_2}{\partial t_1} = -4\Delta t_1 \int_0^1 \frac{\tau(1-\tau)^2}{6} d\tau,$$

$$\frac{\partial F_2}{\partial t_2} = 4(\Delta t_1) \int_0^1 \frac{\tau(1-\tau)^2}{6} d\tau + 4(\Delta t_2) \int_0^1 \frac{\tau(1-\tau)^2}{6} d\tau,$$

$$\frac{\partial F_1}{\partial t} = -4(\Delta t_1) \int_0^1 \frac{\tau(1-\tau)^2}{6} d\tau,$$

$$\frac{\partial F_1}{\partial t_1} = 4(\Delta t_0) \int_0^1 \frac{\tau(1-\tau)^2}{12} d\tau + 4(\Delta t_1) \int_0^1 \frac{\tau(1-\tau)^2}{6} d\tau.$$

Now

$$\int_0^1 \frac{(1-\tau)^2}{6} d\tau = 1/72$$

and

$$\int_0^1 \frac{\tau(1-\tau^2)d\tau}{12} = 1/48$$

so that

$$a_{21} = -\Delta t_1/18,$$

$$a_{12} = -\Delta t_1/18,$$

$$a_{22} = \frac{\Delta t_1 + \Delta t_2}{18}, \quad a_{11} = \frac{\Delta t_0}{12} + \frac{\Delta t_1}{18}.$$

Setting (2.3) and (2.4) equal to zero, we get $\Delta t_2 = \Delta t_1$, $\Delta t_1 = \sqrt{3/2} \Delta t_0$ and thus $a_{22} = 2\Delta t_1/18$, $a_{11} \geq \frac{11/5 \Delta t_1}{18}$ and so $a_{21}a_{12}/a_{11}a_{22} \leq 1/22/5 < 1/4$ so that the proposition applies and the determinant is positive. The rest of the proof in [1] applies directly and thus our assertion is proved.

Finally we consider the case where

$$S_N^{2k} = \{\text{splines of order } 2k \text{ with } N \text{ interior knots each with multiplicity } 2k-1\}.$$

Here, our remainder formula has the form

$$(2.5) \quad R(f) = \int_0^1 f^{(2k)}(s) e \equiv \int_0^1 f^{(2k)}(w-s) |$$

where here our weight is assumed to be in $C^{2k}[0,1]$ and $s \in S_N^{2k}$. Upon integrating by parts $2k$ times, one obtains

$$(2.6) \quad B_0(f) + \dots + B_{2k-1}(f) + \sum_{i=1}^N \sum_{j=0}^{2k-2} (-1)^{j+1} A_{ij} f^{(j)}(t_i) + \int_0^1 w^{2k} f dt$$

where $B_\ell(f)$ involves boundary values of the ℓ th derivative of f and $A_{ij} = s^{(j)} \Big|_{t_i^+}^{t_i^-}$. For these type quadrature formulae and under assumptions analogous to those of Theorems 1 and 2 of this section we again obtain unique and eventually unique optimal quadrature formulae. That is

Theorem 3. Let $w^{(2k)} > 0$ and $\log w^{(2k)}$ concave on $[0,1]$. Then there is a unique optimal quadrature formula of type (2.6) minimizing $\sup\{|R(f)| : \int_0^1 [f^{(2k)}]^2 \leq 1\}$. If no assumption is made on $\log w^{(2k)}$ but $w \in C^{2k+3}[0,1]$

then for large enough N there is a unique optimal quadrature formula of type (2.6).

References

1. D. L. Barrow, C. K. Chui, P. W. Smith, J. D. Ward, Unicity of Best Mean Approximation by Second Order Splines with Variable Knots, in manuscript and announced in Bull. Amer. Math. Soc. as "Unicity of best L_2 approximation by second-order splines with variable knots", Vol. 83, (1977), to appear.
2. C. K. Chui, P.W. Smith, J. Chow, Best L_2 approximation from nonlinear spline manifolds I - Unicity Results, in manuscript.
3. C. K. Chui, P. W. Smith, J. D. Ward, On the smoothness of best L_2 approximants from nonlinear spline manifolds, Math. Comp. 31 (1977), 17-23.
4. I. J. Schoenberg, Monosplines and Quadrature Formulae, in "Theory and Applications of Spline Functions" (T. N. E. Greville, Ed.), Academic Press, N. Y. (1960), 157-207.

NL9 - AN ADAPTIVE ROUTINE FOR NUMERICAL QUADRATURE

Arthur Hausner
Harry Diamond Laboratories
Management Information Systems
Adelphi, Maryland 20783

ABSTRACT. NL9 (Near Lobatto, 9-point) is an adaptive routine that provides an estimate of the definite integral

$$y = \int_a^b f(x) dx$$

to within an absolute or relative error supplied by the user. The basic scheme compares successive estimates of subintegrals after applying 1-, 3-, and 5-point near-Lobatto quadrature formulas. These formulas differ from true Lobatto formulas in that the transformed near end points of ± 0.9999 are used instead of ± 1 for function evaluations, in order to accommodate integrands with end-point singularities. If the estimate proves unsatisfactory, an optimum 9-point Gaussian formula is applied, using the previous 5 points. If the integral estimate is still unsatisfactory, interval bisection and queue-stacking is implemented, with the basic scheme applied to the new subintervals. After all subintervals in any one cycle are processed, an Aitken δ^2 transformation attempts to accelerate the sequence of estimates obtained by successive cycles. This procedure is very effective for increasing efficiency in problems with end-point singularities, without affecting reliability. The queue is currently set for 16 subintervals; when the queue is filled, NL9 switches to a 96-point Gauss-Legendre quadrature formula with queue-stacking and cycle acceleration. This was found to increase efficiency for highly oscillatory integrands without affecting reliability.

NL9 has been tested with a wide variety of integrands and error specifications, and has been found to be an efficient, reliable routine, suitable for general-purpose applications by unsophisticated users. Unlike its predecessor, FOGIE, NL9 handles discontinuous integrands with good reliability and efficiency. In addition, the efficiency of NL9 has been improved overall (the minimum number of function evaluations is 3) at only a slight expense in reliability.

1. INTRODUCTION. NL9 (Near-Lobatto, 9-point) is an adaptive routine written in FORTRAN that provides an estimate of the definite integral

$$(1) \quad Y = \int_a^b f(x) dx$$

to within an absolute or relative error supplied by the user.

It was written to overcome two main deficiencies of its predecessor, FOGIE [1]:

- a. Unreliability for discontinuous integrands.
- b. Inefficiency for low accuracies because the minimum number of function evaluations was 24.

Of course, the large minimum number of function evaluations resulted in FOGIE being an extremely reliable code, except for discontinuous integrands. For use as a general-purpose quadrature routine for unsophisticated users, however, it is desirable to have a routine that will return correct estimates with high probability for all types of integrands, and do it in an efficient way, i.e., with a comparatively small number of function evaluations. The main reason for the unreliability of FOGIE for discontinuous integrands was that no function evaluations were made at the end points of any subinterval. To overcome this for a FOGIE-like routine, Lobatto formulas should be used, whereby function evaluations are required at ± 1 of the transformed interval (the end points). However, if this were done, some integrands with end-point singularities, which blow up at the end points, would have to be specially treated. It was therefore decided at the outset that near-Lobatto formulas would be used, requiring function evaluations at ± 0.9999 of the transformed interval. The loss in reliability due to this change is negligible [1].

To overcome the second disadvantage of FOGIE, a different scheme was implemented for removing a subinterval from the queue of subintervals to be processed. As will be seen, this resulted in the possibility of removing a subinterval from the queue after only three function evaluations. Since the initial interval is the first subinterval, the minimum number of function evaluations was reduced to three. This reduction does decrease reliability, but not to the extent that one may suppose. The major difficulty is in bypassing a narrow peak that contributes to the integral.

This paper discusses various features of NL9 and the tests that were made, leading to the conclusion that it is a very reliable and highly efficient routine for numerical quadrature--particularly suited as a general-purpose routine for unsophisticated users.

2. THE BASIC SCHEME. The basic scheme of NL9 can be conveniently broken into four sections best understood by first describing the calling sequence of the routine (REAL*4 is single precision and REAL*8 is double precision):

```
CALL NL9 (FCT,XL,XU,ACC,INTVL,LIMIT,Y,ERROR,NFUN,IER)
```


FCT = the dummy name of the subroutine to compute $f(x)$.
 XL = the lower limit of integration (real*8).
 XU = the upper limit of integration (real*8).
 ACC = the accuracy desired (real*4).
 If $ACC > 0$, ACC is used for the absolute error.
 If $ACC < 0$, -ACC is used for the relative error.
 INTVL = the number of equal intervals into which (XL,XU)
 is initially divided. The integral of each interval
 is found separately and the results are summed.
 If $INTVL < 0$, -INTVL is the subdivision number and a 96-point
 Gauss-Legendre formula is used as the method of integral
 estimation.
 LIMIT = the maximum number of function evaluations allowed.
 Y = the returned integral estimate (real*8).
 ERROR = if positive, the returned magnitude of
 the estimate of error (real*4). If negative, -ERROR is
 the magnitude of error estimate and the value Y was obtained
 by acceleration.
 NFUN = the number of function evaluations.
 IER = the output error code.
 =1, the answer is thought to be within the accuracy specified.
 =2, the error is thought to exceed that which was specified
 by ACC.
 =3, the run was aborted because of exceeding LIMIT.
 =4, the run was aborted because the queue was filled while
 using the 96-point Gaussian formulas.

2.1 The Initial Phase. We assume $INTVL=1$, so that near-Lobatto formulas
 will be applied to the entire interval in the initial phase. The interval is
 transformed to $(-1,+1)$ for simplicity in applying the formulas. Estimates
 are then obtained generally by using

$$(2) \quad I_n = \sum_{i=1}^n w_i f(x_i)$$

where the w_i and x_i depend on n . For $n=1$, a 1-point estimate has

$$(3) \quad x_1 = 0, \quad w_1 = 2.$$

The estimate I_1 is exact for polynomials to degree 1.

For $n=3$, we use

$$(4) \quad \begin{aligned} x_1 &= 0, & w_1 &= 1.3331999799973329995 \\ x_{2,3} &= \pm 0.9999, & w_{2,3} &= 0.33340001000133350002. \end{aligned}$$

The estimate I_3 is exact for polynomials to degree 3. I_1 is compared to I_3 (see section 3) and the process continues if the criteria for stopping are not satisfied.

I_5 is next computed from

$$(5) \quad \begin{aligned} x_1 &= 0, & w_1 &= 0.71103996442309754558 \\ x_{2,3} &= \pm 0.9999, & w_{2,3} &= 0.10009004501475245572 \\ x_{4,5} &= \pm 0.65458817259184819938, & w_{4,5} &= 0.54438997277369877148 \end{aligned}$$

which computes integrals of polynomials to degree 7 exactly. I_3 is compared to I_5 (see section 3) and the process continues if the criteria for stopping are not satisfied.

I_9 is then computed with

$$(6) \quad \begin{aligned} x_1 &= 0, & w_1 &= 0.34372776695800701539 \\ x_{2,3} &= \pm 0.9999, & w_{2,3} &= 0.030740824922769000059 \\ x_{4,5} &= \pm 0.65458817259184819938, & w_{4,5} &= 0.28395029242094264548 \\ x_{6,7} &= \pm 0.89031636893046580368, & w_{6,7} &= 0.17924470129867993407 \\ x_{8,9} &= \pm 0.34094819564600667896, & w_{8,9} &= 0.33420029787860491270 \end{aligned}$$

which computes exact estimates of integrals of polynomials to degree 13. I_5 and I_9 are compared (see section 3) and the process continues if the criteria for stopping are still unsatisfied.

In the above, only the 3-, and 5-point estimates are near Lobatto. The 9-point estimate uses the previous five function evaluations.

Throughout the initial phase, a check is made for $f(x_i) \approx f(x_{i+1})$ for $1 \leq i \leq 4$. If all checks are true to 11 digits or more, the function $f(x)$ is flagged symmetric (even), and it is therefore assumed that $f(x) = f(-x)$,

x being the coordinate in the transformed interval. This results in a savings of half of the function evaluations thereafter. The number of function evaluations saved, N_s , can always be computed from the returned NFUN:

$$(7) \quad N_s = \left([NFUN-9] - 2 \times \left[\frac{NFUN-9}{2} \right] \right) (NFUN-9).$$

2.2 Queue Stacking. If the initial phase does not result in exiting from the routine, the 9-point estimate and its upper and lower bounds are placed in a circular stacking queue. Subintervals on this queue are processed by subdividing them into two equal intervals and applying to each half the basic scheme described in 2.1. It is thus possible that none, one, or both halves of a subinterval are placed back on the queue, with the original one being removed. Additional tests compare the sum of the estimates of each half with that of the whole. If any subinterval passes an accuracy test, the estimate of its integral is accumulated in the variable SUMY in this routine, and is not placed back on the queue.

If the queue is empty, SUMY is returned as the final estimate Y. Otherwise, the process continues until LIMIT is reached, the queue is filled, or estimates are found from the acceleration procedure.

2.3 Acceleration Procedure. The routine knows when the set an entire set of subintervals on the queue has been processed, i.e., the set has been subdivided and new estimates have been found. This is called the end of a cycle. The sum of the new estimates plus SUMY constitutes a new total estimate of the integral at the end of a cycle. These estimates form a sequence which is amenable to acceleration. The identical acceleration algorithm as described for FOGIE [1] is used here, consisting primarily of repeated applications of Aitken's δ -squared transformation. Such a procedure has been found to be very useful for accelerating these sequences to their limits for cases that have end-point singularities. If the acceleration criteria are satisfied, NL9 returns with an answer even if intervals still remain on the queue.

2.4 Switching to a 96-point Gaussian Formula. If the queue becomes filled (16 entries), it is assumed that the integral contains high frequencies, so that the low-order formulas used are not efficient. Contiguous subintervals are combined and replaced on the queue and, thereafter, only a 96-point Gauss-Legendre formula is applied in all cases. The best estimate obtained with the 9-point formulas is stored for comparison with that of the 96-point formula at the end of the next cycle. If acceleration is performed, it is with estimates obtained only with 96-point formulas.

If the queue becomes filled while the 96-point formula is being used, NL9 is aborted. Either the integral does not exist, or such high frequencies are present that INTVL<-1 should be used.

3. CRITERIA FOR REMOVAL FROM QUEUE. Four possible tests are used to determine whether a subinterval should be removed from the queue:

$$1. \text{ If } |I_3 - I_1| \leq |I_3| * 10^{-15} \text{ or } \quad (\text{bypassed when NFUN}=3, \text{ if } I_3=0 \text{ or there was underflow})$$

$$(8) \quad |I_5 - I_3| \leq |I_5| * 10^{-15} \text{ or}$$

$$|I_9 - I_5| \leq |I_9| * 10^{-15}$$

2. For NFUN>9, and with Y the best estimate to date,

$$|I_3| \leq \text{ACC}/10^3 \quad (\text{ACC}>0) \text{ or } \quad (\text{bypassed if the function has never oscillated})$$

$$(9) \quad |I_5| \text{ or } |I_9| \leq \text{ACC}|Y|/10^2$$

$$|I_3| \leq -\text{ACC} * |Y|/10^4 \quad (\text{ACC}<0) \text{ or}$$

$$|I_5| \text{ or } |I_9| \leq -\text{ACC} * |Y|/10^3$$

$$3. \text{ If } |I_3 - I_5| \leq \text{ACC}/\text{EK} \quad (\text{ACC}>0)$$

$$\text{or } |I_5 - I_9| \leq \text{ACC}/\text{EK}$$

$$(10) \quad \text{or}$$

$$\text{If } |I_3 - I_5| \leq -\text{ACC}|Y|/\text{EK} \quad (\text{ACC}<0)$$

$$\text{or } |I_5 - I_9| \leq -\text{ACC}|Y|/\text{EK}$$

where

$$(11) \quad \begin{aligned} \text{EK} &= \infty \text{ if } |I_3 - I_5| > 0.1 |I_5| \text{ or } |I_5 - I_9| > 0.1 |I_9| \\ \text{EK} &= 100 \text{ if } |I_3 - I_5| > 10^{-3} |I_5| \text{ or } |I_5 - I_9| > 10^{-3} |I_9| \\ \text{EK} &= 10 \text{ if } |I_3 - I_5| > 10^{-5} |I_5| \text{ or } |I_5 - I_9| > 10^{-5} |I_9| \\ \text{EK} &= 1 \text{ if } 10^{-5} |I_5| \geq |I_3 - I_5| \text{ or } 10^{-5} |I_9| \geq |I_5 - I_9| \end{aligned}$$

Thus, this condition depends on the relative error of successive estimates.

4. When each component is tested after subdivision for tests 1,2, and 3, the sum of 9-point estimates for the left- and right-hand sides forms an 18-point estimate I_{18} . This estimate is then compared to the 9-point estimate of the original subinterval. An identical test to 3 is made, with I_{18} replacing I_5 where it appears in both (10) and (11).

These tests are performed as the various quantities I_1, I_3, I_5, I_9 , and I_{18} are found. If any subinterval is removed from the queue, of course, subsequent tests are bypassed.

If a 96-point formula is used, tests 1 and 4 are made with $EK = 10$.

These tests determine whether or not any subinterval is put back in queue. If the queue becomes empty, control is returned to the calling program.

4. END-OF-CYCLE TESTS. At the end of each cycle of subdividing, the best estimate K_n is compared to the best estimate K_{n-1} of the previous cycle. If

$$(13) \quad 10 * |K_n - K_{n-1}| \leq ACC \text{ if } ACC > 0$$

$$\text{or } 10 * |K_n - K_{n-1}| \leq -ACC * |K_n| \text{ if } ACC < 0$$

(provided $|K_n - K_{n-1}| \leq 0.1 |K_n|$, if no oscillations are present) then K_n is adjudged to pass the test and is returned as the answer. If it does not pass the test, the sequence of K_j , $j=1, n$, are transformed into a new sequence K'_j , $j=3, n$ by means of the Aitken δ^2 transformation as in FOGIE [1], and the test (13) is applied to the values K'_n and K'_{n-1} . Repeated transformations are obtained and repeated tests are made as long as there are at least three elements left after the first transformation. The acceleration is bypassed when $n < 4$ initially, or if the difference sequence is not monotonic decreasing in magnitude. In the latter case, only the two best values are retained, with K_2 set to K_n and K_1 to K_{n-1} , and a new sequence is begun. Acceleration may then be attempted after two more cycles of subdividing.

5. ERROR ESTIMATES. Each time a subinterval is removed from the queue, it introduces an error, the sums of the squares of which are accumulated. The magnitude of the component of error is adjusted empirically, depending upon the method by which it was removed from the queue. For test number 1, the magnitude is taken to be the left-hand quantities of (8) multiplied by 10. For other tests, the multiplying factor is EK . The final error is taken to be the square root of the accumulated sum of squares of these components, modified in an attempt to prevent underestimating the error. If its value already exceeds ACC (or $-ACC * |Y|$ if $ACC < 0$) the IER is returned with the value 2. Otherwise, the error estimate is reset to

$$(14) \quad ERROR = MIN(20 * ERROR, ACC)$$

Although empirical, the estimate worked very well in almost all cases tested. The actual value is not too important. Keeping track of its value internally permits the return of IER = 2, if the user requests an absurd accuracy requirement. Because test number 1 will insure an empty queue eventually, no matter what accuracy is specified, the main use of ERROR is to alert the user to these situations.

6. TEST AND RESULTS. A wide variety of test integrals were used to tune the various constants required. Some of those used, in addition, were selected because test results with them have been published, thus providing some means of comparing NL9 with other routines. All tests used INTVL=1 and LIMIT=10000.

The tests consisted of three groups of specific integrals:

1. 21 integrals used by Kahaner [2]. These were tested with $ACC = +10^{-3}, +10^{-6}, +10^{-9}, +10^{-12}$. Comparisons are available with some other routines for $ACC = 10^{-3}, 10^{-6}$, and 10^{-9} .
2. 50 integrals used by Casaletto, Picket, and Rice, and published in [3]. These were tested for $ACC = +10^{-1}$ through $+10^{-9}$. Comparisons are available only with the routine FOGIE.
3. 15 integrals mentioned specifically by other papers in the field ([4] through [6]), or included just to see what happened. They are listed in Appendix A, and were tested for $ACC = +10^{-1}$ through $+10^{-9}$.

In addition to these tests, some parameter sweeping tests, described by de Boor [5], were made for $ACC = +10^{-6}$:

$$(a) \quad \int_0^1 x^\alpha dx \quad \text{for} \quad \alpha = -\frac{31}{32}(\frac{1}{32})^2$$

$$(b) \quad \int_{-1}^1 \frac{2^{-\alpha}}{2^{-2\alpha} + x^2} dx \quad \text{for} \quad \alpha = 0(1)100$$

$$(c) \quad \int_0^1 [1 + \cos(\alpha\pi x)] dx \quad \text{for} \quad \alpha = 0(\frac{1}{3})\frac{250}{3}$$

$$(d) \quad \int_0^1 (-\ln x)^\alpha dx \quad \text{for} \quad \alpha = -0.99(0.91)2$$

$$(e) \quad \int_0^1 U(\alpha-x) dx \quad \text{for} \quad \alpha = 0.01(0.01)1$$

Tests a to c can be compared to CADRE for $ACC = 10^{-6}$, and tests a through d can be compared to FOGIE for $ACC = 10^{-6}$.

Results for the three groups of specific integrals are summarized in Tables 1 through 5.

For Kahaner's test integrals, failures occurred only with case 21, which caused failures for most routines Kahaner tested. Comparisons can be made with CADRE (easily the "winner" of the Romberg extrapolation routines with results of this test published) for $ACC = 10^{-3}$, 10^{-6} , and 10^{-9} , and FOGIE for those same accuracies. Counting a failure for each routine for the same case as a tie, and the number of function evaluations as the measure of efficiency, Table 6 summarizes the comparisons. The relaxation of conditions for queue removal for NL9 permitted three failures as compared to zero for FOGIE, but the efficiency was markedly improved. NL9 is comparable to CADRE for the low accuracy (10^{-3}) but appears more efficient for the higher accuracies (10^{-6} and 10^{-9}). FOGIE still seems most efficient for the high accuracy of 10^{-9} , where the minimum number of function evaluations of 24 is not detrimental.

Tables 2 and 3 show results for the test integrals of Casaletto, Picket, and Rice. Complete failure occurs for case 47, where a very narrow discontinuous band is superimposed on a low-degree polynomial. Most routines fail on this case because no points are sampled in the discontinuous region early enough. (FOGIE does sample points in this region, but fails on some of these cases because of its unreliability for discontinuous cases.) It is clear, however, that NL9 is vastly superior to the Clenshaw-Curtis algorithm described by Gentleman [3] from the qualitative description of the results of those tests. There were many failures as well as many cases (end-point singularities and discontinuous) which could not be computed by the algorithm in 10000 function evaluations. The maximum required by NL9 was 2103. Further, their statistical test on the "smooth" cases shows NL9 is more efficient for accuracies greater than 10^{-3} . No comparisons are available for relative error tests, where failure still occurs with case 47. Also, as expected, case 50 always returned $IER = 2$. Since the answer is 0, no result can meet this requirement. This is a failure in a sense, but was not so indicated.

Few comparisons are available with the test integrals in Group 3, summarized in Tables 4 and 5. The only failure with absolute error occurs with case 11, $ACC = 10^{-9}$. The internal point of singularity is actually hit exactly (64 bits) and causes overflow after many interval bisections. The results shown is when $f(x) = 0$ at $x = 0.2$ was used to prevent overflow. For the relative error runs (Table 5), this case also presents difficulty. $IER = 4$ is returned, indicating

a filled queue. Case 6 had many runs returned with IER = 2, although the accuracy was met. This results from loss of digits due to summing nearly equal positive and negative terms. One can expect IER = 2 in such cases when relative error is used. The overall results of these test cases indicates both excellent reliability and efficiency.

Figures 1 through 10 show the test results for cases where parameters were swept. The graphs are paired for each case -- one showing the number of functions evaluation and one the true error. Only results for $ACC = 10^{-6}$ are given because they can be compared, in part, to both FOGIE and CADRE. Results for $ACC = -10^{-6}$ will be described if they are significantly different. Downward or upward arrows indicate cases out of range of the scale.

Figures 1 and 2 are for case a, the powers of x . No failures were obtained in the entire range (for relative errors also), with the average number of function evaluations slightly less than FOGIE and CADRE. (CADRE fails for some cases.) All of the cases where $NFUN > 63$ returned successfully as the result of the acceleration process. This case can be considered completely successful.

Figures 3 and 4 are for case b, the large central peak. Again, there were no failures with the true error about constant. The number of function evaluations is about one-third that of FOGIE and slightly less than CADRE (CADRE fails for $\alpha > 30$). Some of that difference results from the integrand being a symmetric function. The results for relative error are similar, but NFUN was greater by more than a factor of 2. This occurred because the large value of the peak is not apparent until many cycles of subdividing and tends to keep intervals on the queue more than necessary. This case can also be considered completely successful.

Figures 5 and 6 are for case c, the oscillatory integrand. There were no failures, with absolute and relative error test being identical, a reasonable result since the answer is always near the value 1. Results are somewhat superior to FOGIE, because of the switch to the 96-point Gauss-Legendre formulas for a filled queue for the larger frequencies. They are vastly superior to CADRE, which fails in many cases because it concludes the integrand is linear. Note that NFUN was quite small for many values of large α . For integral α , the integrand can become anti-symmetric after some subdivision. NL9 recognizes this situation and stops more quickly than it would for nonintegers near α . This case also can be considered completely successful.

Figures 7 and 8 show results for case d, the nonalgebraic end-point singularity. This case has singularities at both ends when $\alpha < 0$ and is considerably more difficult to compute numerically. The true error

graph indicates 12 failures near $\alpha = -1$ (FOGIE had 2) with the arrows for runs that returned IER = 4. In the region of failure the correct values of the integrals are greater than 10. Some of these failures had as many as six correct digits. Only eight failures occurred with the relative error specifications, all with IER = 1. At least four digits were correctly obtained by the acceleration process in those runs. This case, while only partially successful, is considered satisfactory because of the difficult nature of the integral.

Figures 9 and 10 are for case e, the jump discontinuity. No failures were obtained, indicating good reliability. No comparisons with other routines are available, but from other cases run, it is evident that the efficiency is not as good as CADRE. Still, the purpose of using near-Lobatto formulas was to improve reliability. It is doubtful that routines based on Gaussian-type formulas can be as efficient as routines using Romberg extrapolation for detecting jump discontinuities.

7. CONCLUSIONS. From the results of tests described here, we conclude that NL9 is an effective routine for computing numerical quadrature. Overall reliability is excellent for all types of integrands encountered in practice. Overall efficiency is quite good, especially for problems with end-point singularities and high accuracy requirements. These assessments suggest that NL9 is suitable as a general-purpose quadrature routine that will satisfy most users.

Ultimately, a routine with keywords -- similar to the control program designed with EISPACK2 [7] -- might be desirable for unsophisticated users. Information about the integrand, such as "smooth," "singularity," "oscillatory," "discontinuous," and "peak" can be useful in selecting a specific routine to get good efficiency. For example, a "smooth" integrand is probably best attacked by Patterson-like formulas [4], whereas one with end-point singularities should contain the capability of subdivision and acceleration. NL9 might represent a procedure to be used if no keywords are supplied, i.e., if the nature of the integrand is unknown. There is then an excellent chance that the answer is correctly obtained in a reasonably efficient manner.

8. REFERENCES.

- [1] A. Hausner and J.D. Hutchison, "FOGIE: An Adaptive Code for Numerical Integrals Using Gaussian Quadrature," ARO Report 75-3, Proceedings of the 1975 Army Numerical Analysis and Computers Conference, pp. 139-177.
- [2] D.K. Kahaner, "Comparison of Numerical Quadrature Formulas", Mathematical Software, John R. Rice, ed., Academic Press, New York, 1971, pp. 229-259.
- [3] W.M. Gentleman, "Implementing Clenshaw-Curtis Quadrature, I Methodology and Experience," Communications of the ACM, Vol. 15, No. 5, May 1972.
- [4] T.N.L. Patterson, "Algorithm 468, Algorithm for Automatic Numerical Integration of a Finite Interval, Communications of the ACM, Vol. 16, No. 11, November 1973.
- [5] C. de Boor, "CADRE: An Algorithm for Numerical Quadrature," Mathematical Software, John R. Rice, ed., Academic Press, New York, 1971, pp. 417-449.
- [6] F.T. Krough and W.V. Snyder, "Preliminary Results with a New Quadrature Subroutine", Computing Memo, No 363, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91103, 30 July 1974.
- [7] J.M. Boyle and A.A. Grau, "Modular Design of a User-Oriented Control Program for EISPACK," Tech Memo TM242, Applied Mathematical Division, Argonne National Laboratory, Argonne, Ill., November, 1973.

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

5 OF 7
ADA
046552



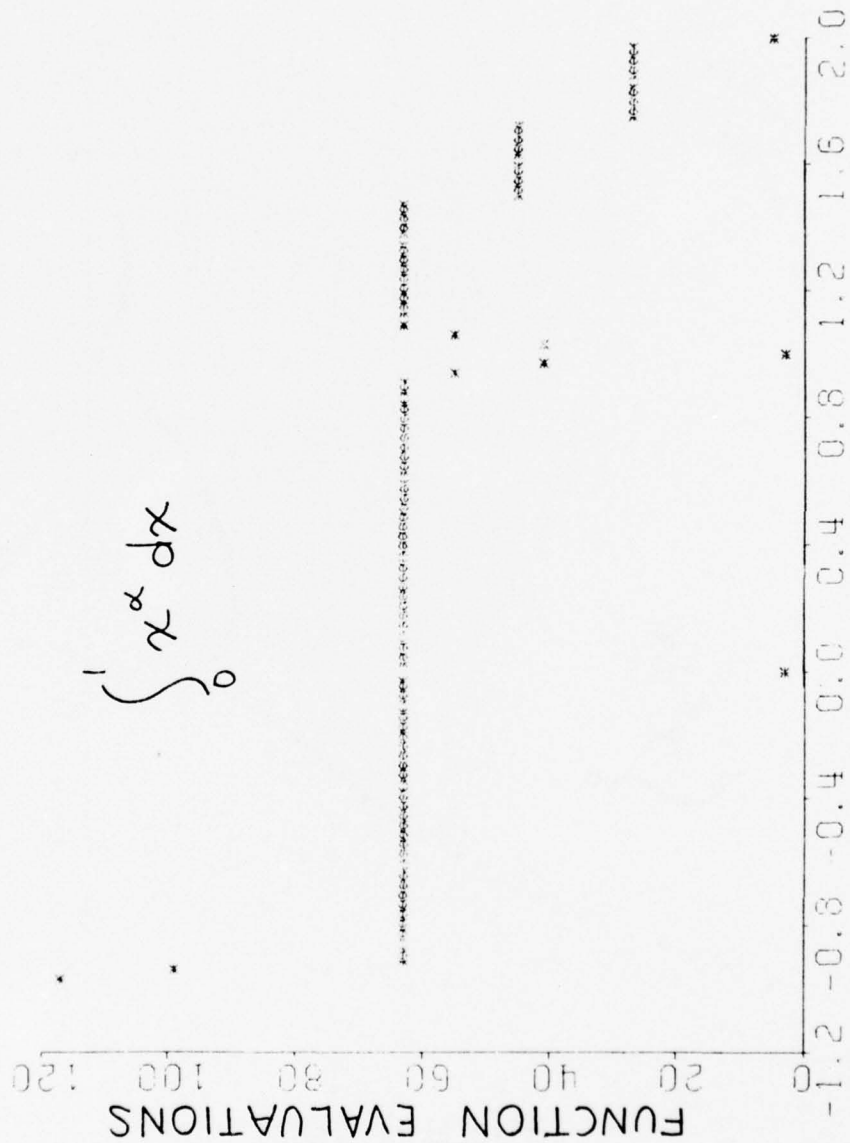


Figure 1. Function evaluations for sweep of powers of x .

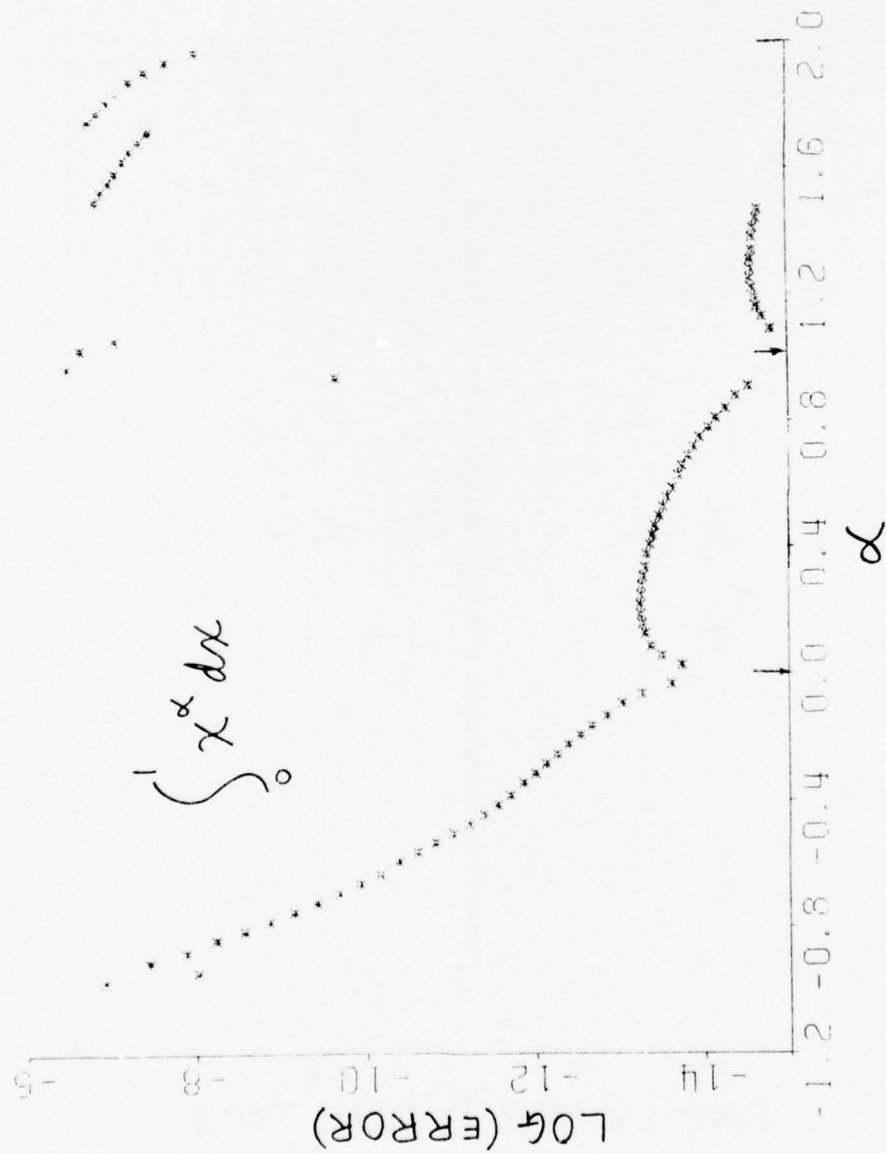


Figure 2. True error for sweep of powers of x .

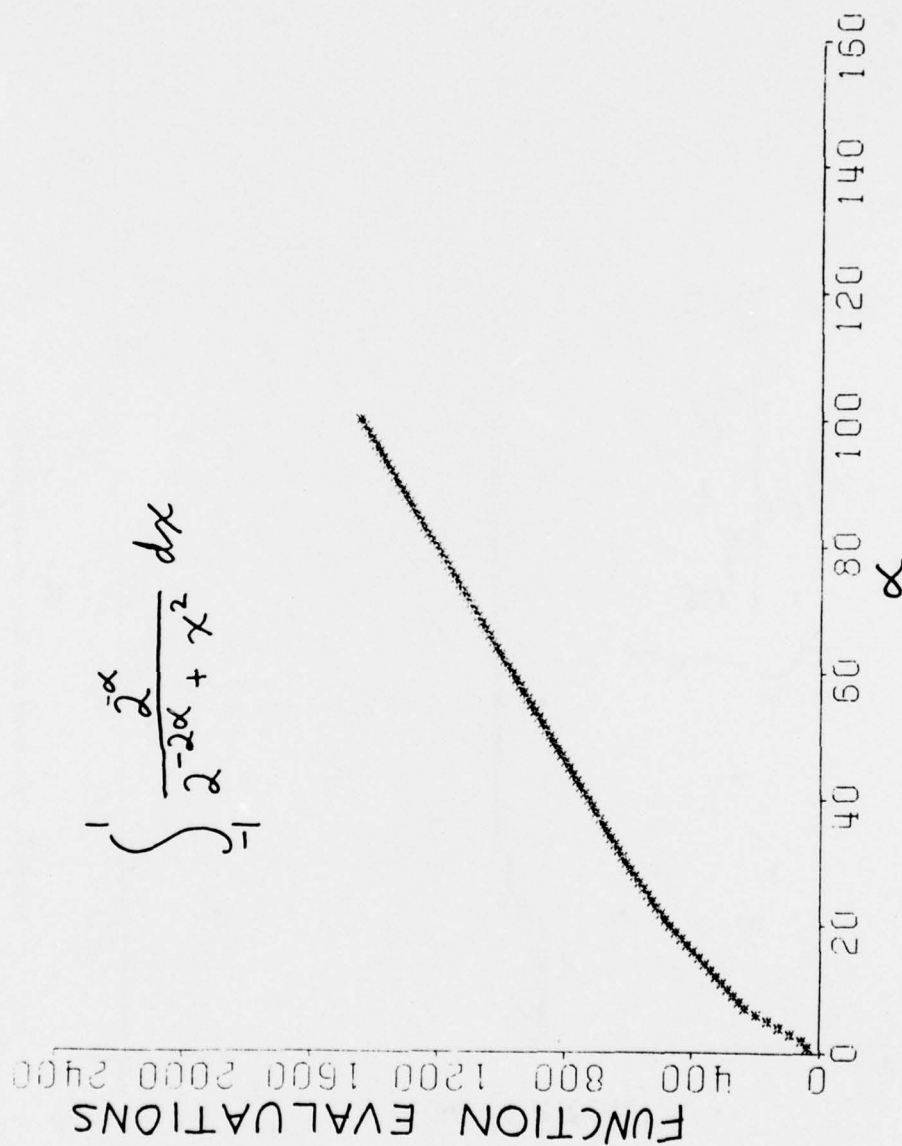


Figure 3. Func' evaluations for sweep of central peak.

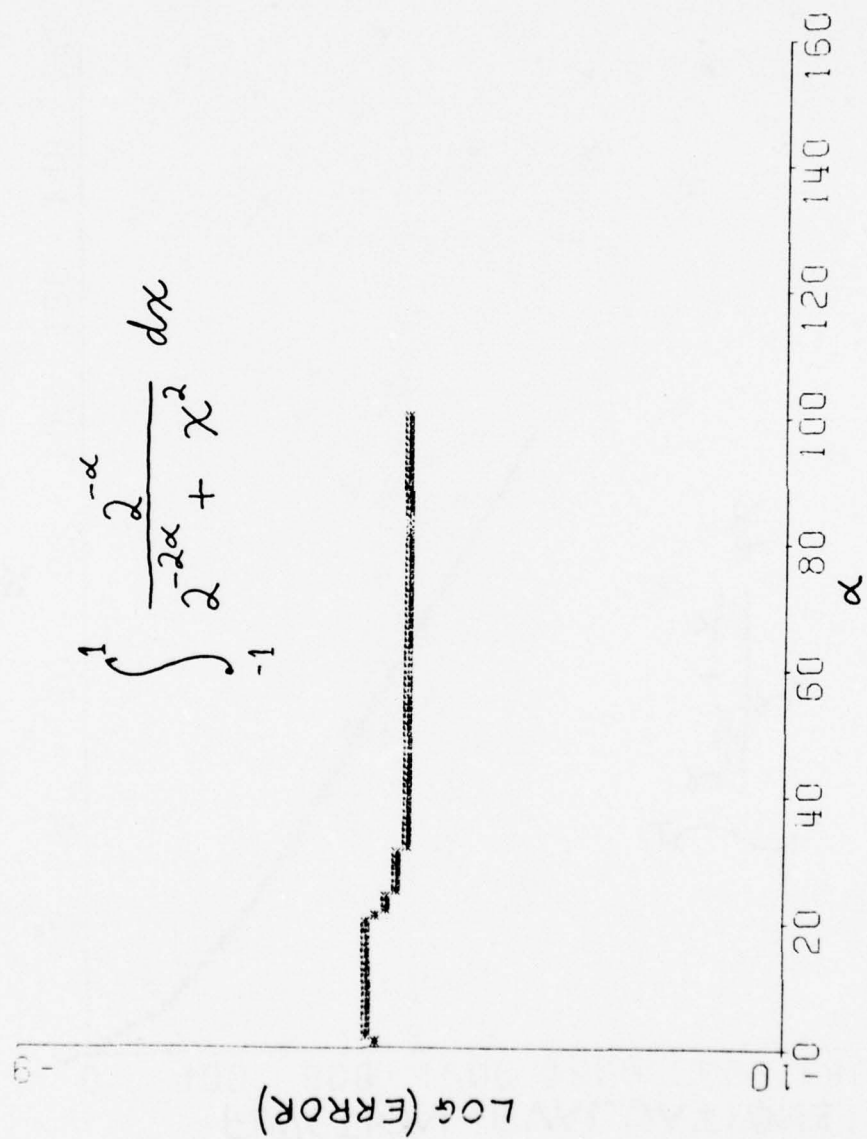


Figure 4. True error for sweep of central peak.

$$\int_{-1}^1 \frac{2^{-2\alpha} + x^2}{2^{-2\alpha} + x^2} dx$$

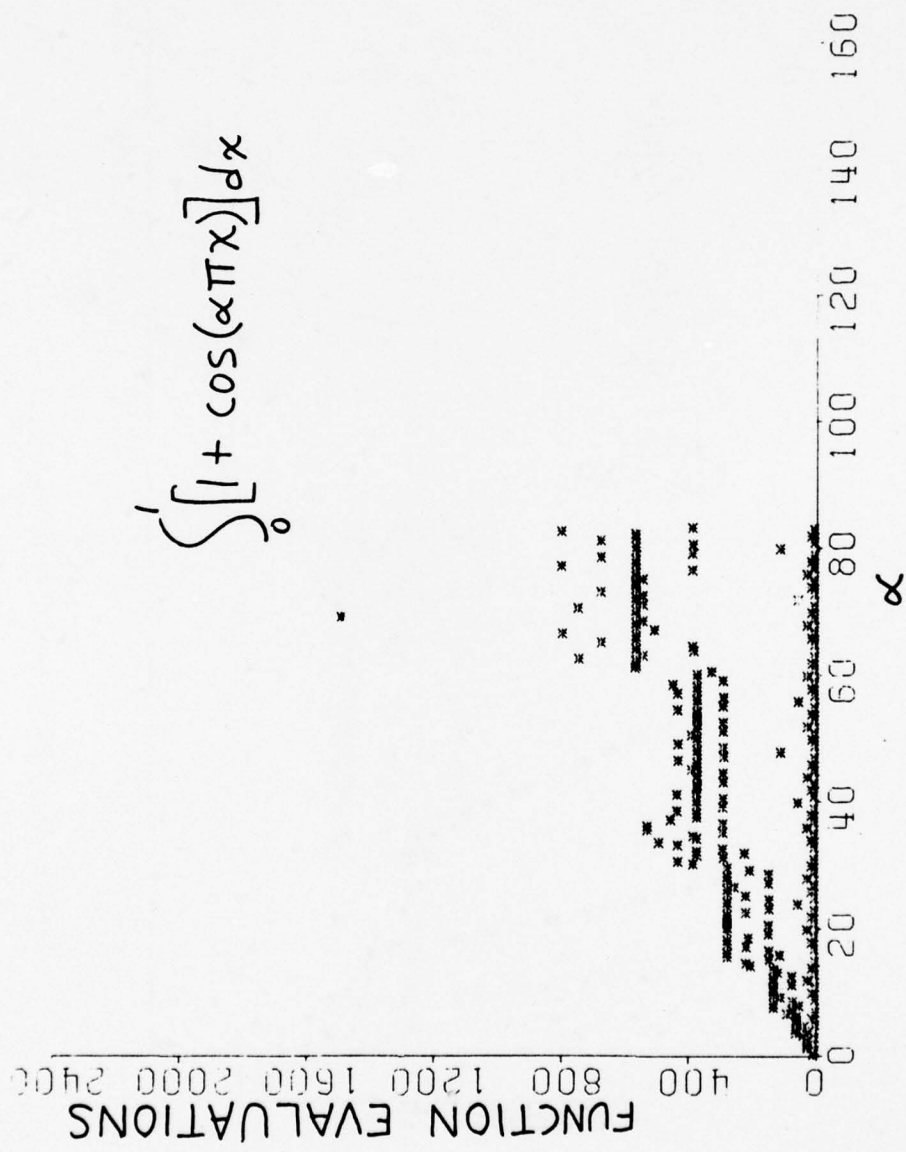


Figure 5. Function evaluations for sweep of oscillatory case.

$$\int_0^1 [1 + \cos(\alpha \pi x)] dx$$

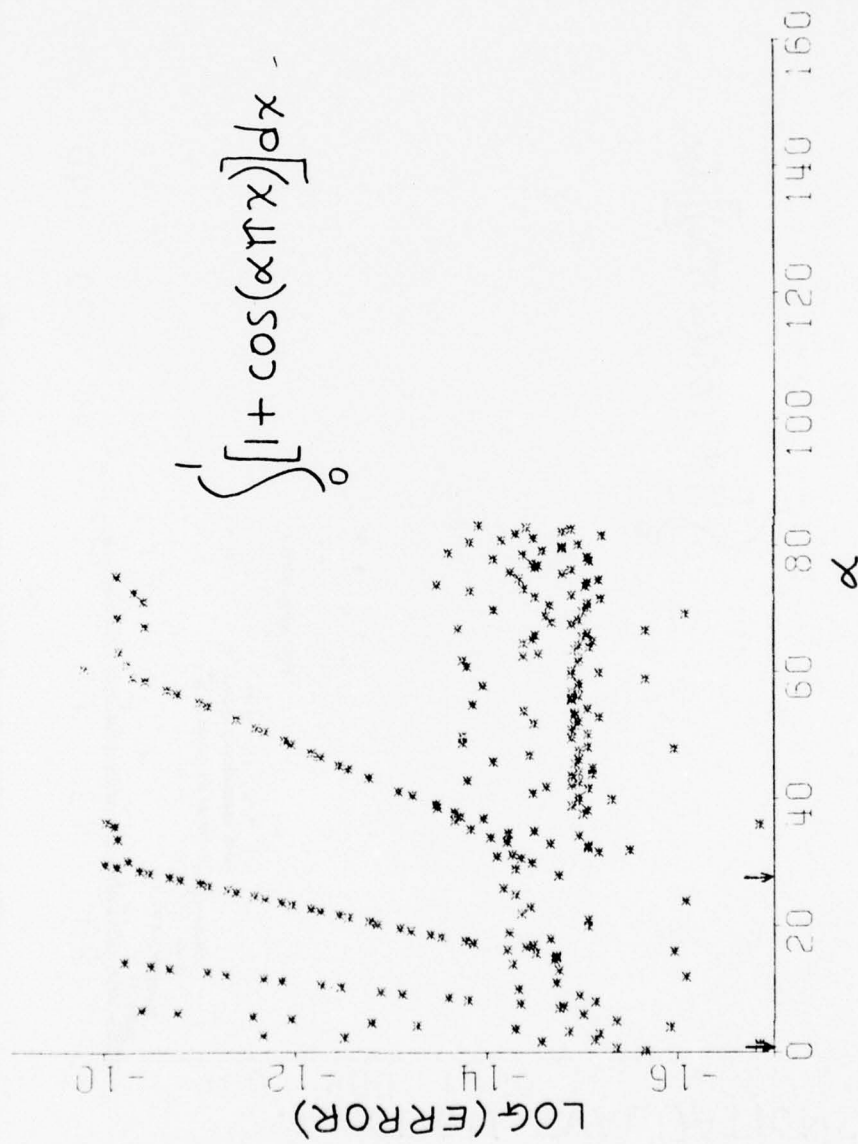


Figure 6. True error for sweep of oscillatory case.

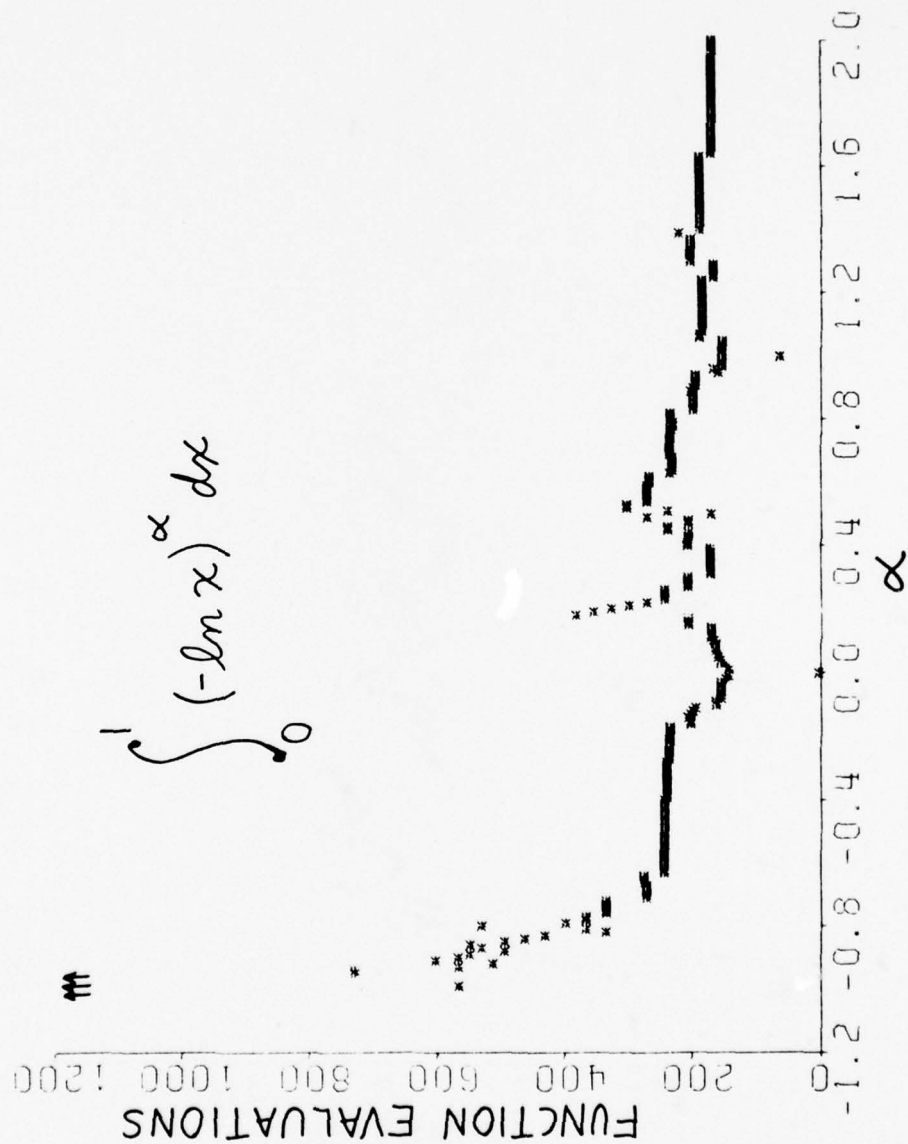


Figure 7. Function evaluations for sweep of nonalgebraic end-point singularity.

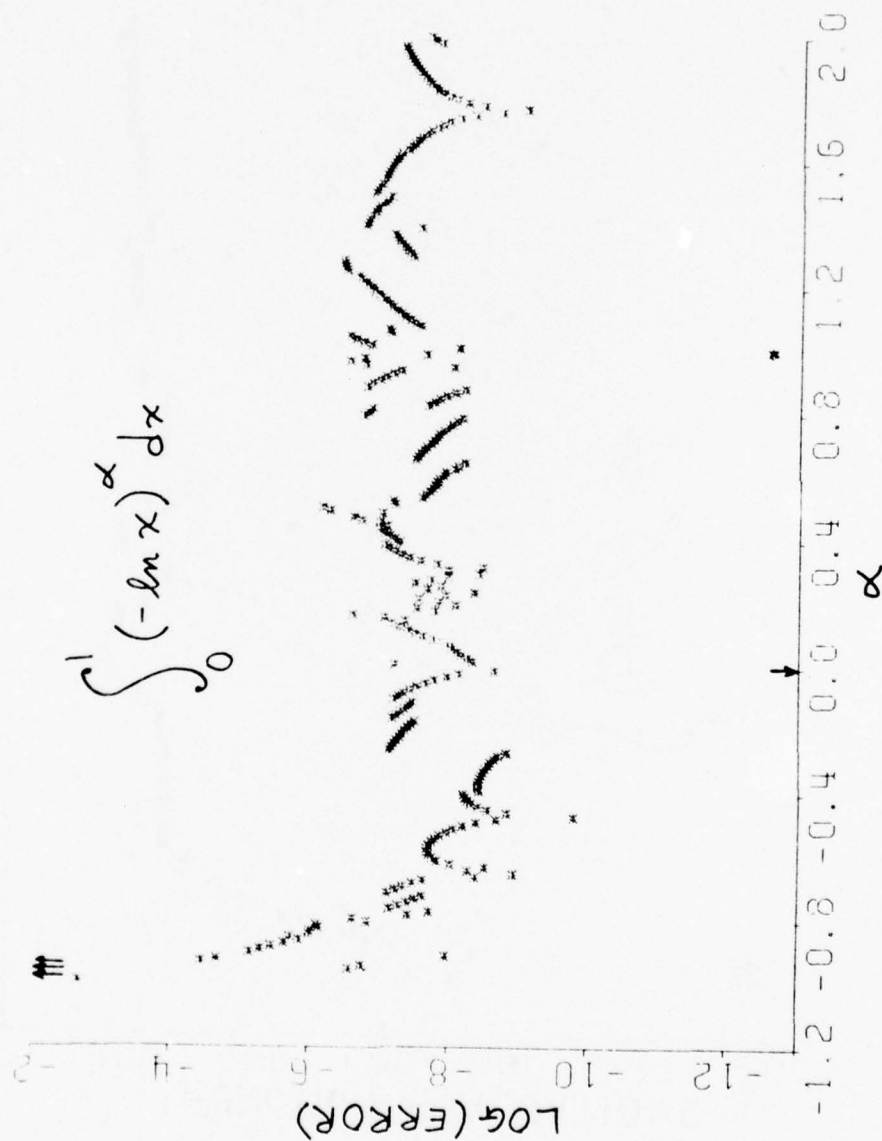


Figure 8. True error for sweep of nonalgebraic end-point singularity.

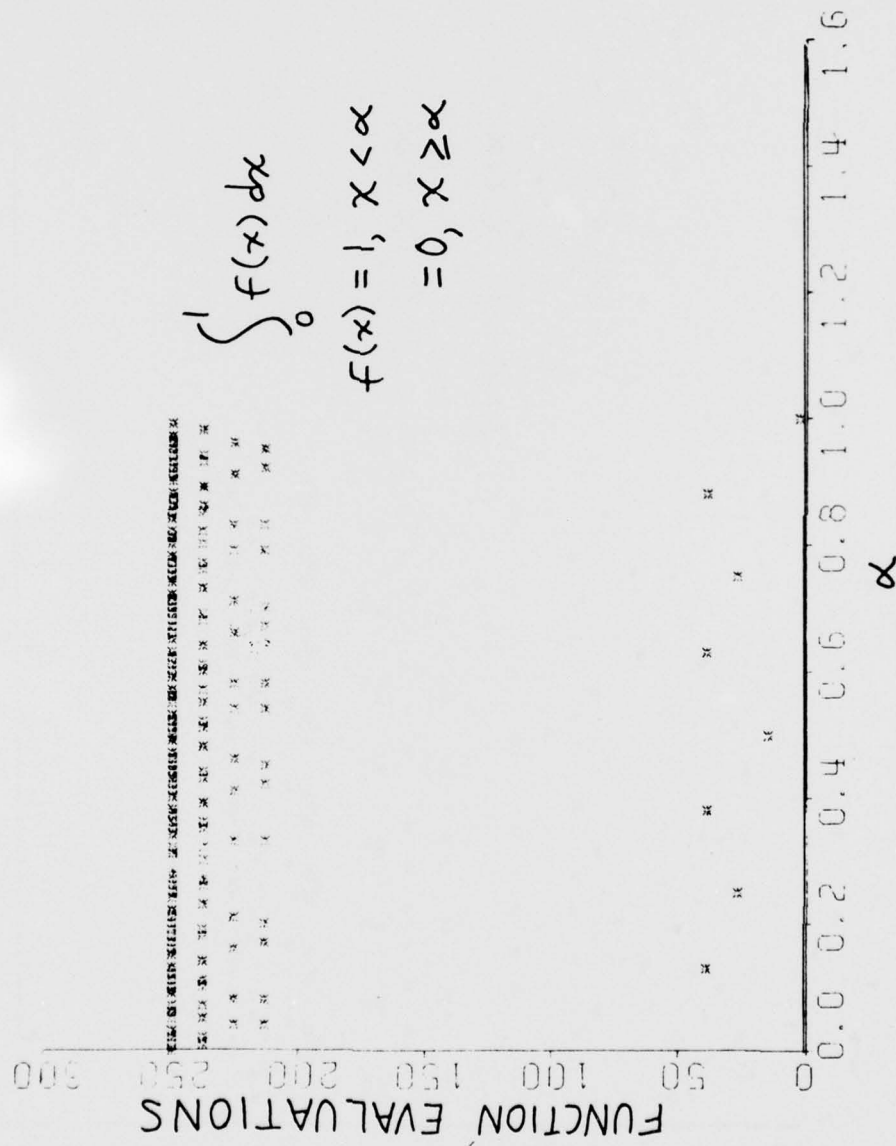


Figure 9. Function evaluations for sweep of jump discontinuity.

$$\int_0^1 f(x) dx$$

$$f(x) = 1, x < \alpha$$

$$= 0, x \geq \alpha$$

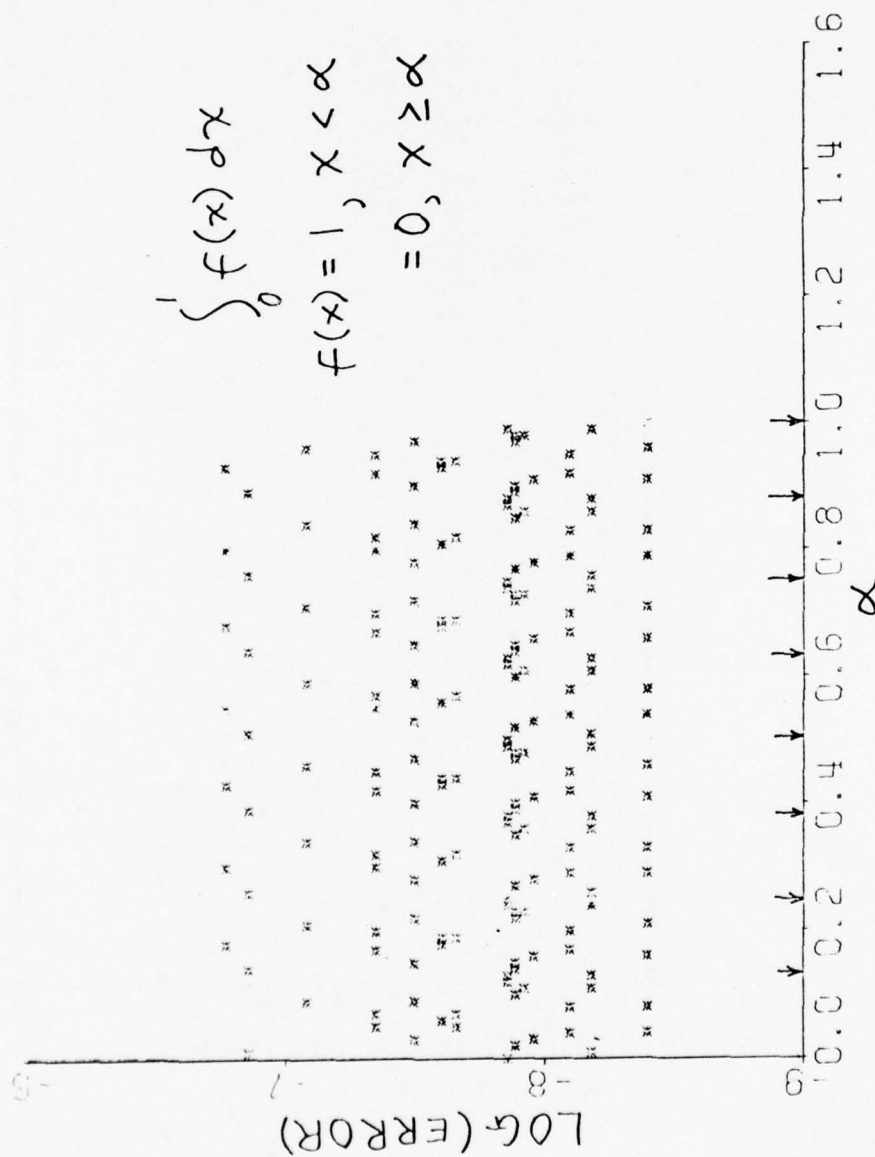


Figure 10. True error for sweep of jump discontinuity.

$$\int_0^1 f(x) dx$$

$$f(x) = 1, x < \alpha$$

$$= 0, x \geq \alpha$$

TABLE 1

RESULTS WITH KAHANER'S INTEGRALS

Value Shown is returned NFUN. F indicates Failure

All returned IER=1

All true errors < estimated errors except for the failures.

CASE \ ACC	10^{-3}	10^{-6}	10^{-9}	10^{-12}	-10^{-3}	-10^{-6}	-10^{-9}	-10^{-12}
1	9	9	27	27	9	9	9	27
2	129	249	369	489	141	261	369	489
3	51	63	99	99	51	63	99	99
4	9	9	18	18	9	9	18	18
5	18	36	36	72	18	36	36	72
6	9	45	81	99	23	63	99	99
7	105	63	99	315	441	63	99	351
8	9	27	63	63	9	27	63	63
9	235	387	951	951	279	459	951	951
10	9	9	27	63	9	27	27	63
11	5	9	27	27	5	9	27	27
12	5	9	9	27	5	9	9	27
13	375	567	567	567	567	567	567	951
14	87	141	155	231	89	153	261	269
15	75	129	171	297	77	147	317	437
16	149	207	351	459	153	297	351	675
17	335	567	567	951	375	567	567	951
18	135	135	279	375	135	207	315	375
19	59	63	99	135	163	63	99	171
20	18	36	36	72	18	18	36	72
21	145F	225F	347F	901	171F	297F	905	1593

TABLE 2 RESULTS WITH ABSOLUTE ERROR FOR INTEGRALS OF CASALETTO,
PICKET, AND RICE

All IER=1. Value shown is NFUN. F indicates Failure.
All true errors < estimated errors except for Failures.

CASE	ACC								
	10 ⁻¹	10 ⁻²	10 ⁻³	10 ⁻⁴	10 ⁻⁵	10 ⁻⁶	10 ⁻⁷	10 ⁻⁸	10 ⁻⁹
1	3	3	3	3	3	3	3	3	3
2	3	3	3	3	3	3	3	3	3
3	5	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5
5	9	9	9	9	9	9	9	9	9
6	5	9	9	9	9	9	9	9	9
7	9	9	9	9	9	9	9	9	9
8	9	9	9	9	9	9	9	9	9
9	9	9	9	9	27	27	27	27	27
10	9	9	9	9	27	27	27	27	27
11	9	9	27	27	27	27	27	27	27
12	9	9	27	27	27	27	27	27	27
13	9	9	27	27	27	27	27	27	27
14	5	9	27	27	27	27	27	27	27
15	9	9	27	27	27	27	27	27	63
16	5	27	27	27	27	27	27	27	63
17	9	27	27	27	27	27	27	63	63
18	5	27	27	27	27	27	27	63	63
19	9	27	27	27	27	27	63	63	63
20	5	27	27	27	27	27	63	63	63
21	5	5	9	9	9	9	9	9	27
22	9	9	9	9	9	18	18	18	18
23	5	5	9	9	9	9	9	9	9
24	5	5	5	9	9	9	9	9	9
25	9	9	9	9	9	27	27	27	27
26	135	163	235	279	379	387	459	951	951
27	9	9	9	27	27	27	27	63	63
28	5	5	5	9	9	9	9	9	27
29	453	375	375	375	567	567	567	567	567
30	777	951	951	951	951	951	951	951	951
31	541	567	567	1335	1335	1335	1335	1335	1719
32	5	5	9	9	9	9	9	18	18
33	18	18	18	18	18	36	36	36	36
34	567	567	567	567	1719	2103	1719	1719	1719
35	9	9	9	9	9	9	27	27	27
36	23	23	51	59	109	63	63	63	99
37	23	37	51	59	63	63	63	63	99
38	23	37	51	55	109	63	63	63	99
39	23	37	51	55	109	63	63	63	99
40	55	55	83	115	135	135	135	135	207
41	9	9	9	27	27	45	63	63	81
42	19	27	27	55	63	95	135	135	135
43	9	9	9	9	27	27	45	45	63
44	9	23	23	27	27	45	63	63	99
45	15	15	15	15	15	15	15	15	15
46	3	3	3	3	3	3	3	3	3
47	5F	5F	5F	5F	5F	5F	5F	5F	5F
48	35	49	89	171	199	263	289	337	365
49	9	9	9	9	9	9	9	9	9
50	3	3	3	3	3	3	3	3	3

TABLE 3 RESULTS WITH RELATIVE ERROR FOR INTEGRALS OF CASALETTO,
PICKET, AND RICE

IER=1 except for superscript values. F indicates Failure.

* Indicates error estimate < true error.

True errors < estimated errors, except for IER=2, Failures,
and (*) indicator.

CASE \ ACC	-10^{-1}	-10^{-2}	-10^{-3}	-10^{-4}	-10^{-5}	-10^{-6}	-10^{-7}	-10^{-8}	-10^{-9}
1	3	3	3	3	3	3	3	3	3
2	3	3	3	3	3	3	3	3	3
3	5	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5
5	9	9	9	9	9	9	9	9	9
6	5	5	9	9	9	9	9	9	9
7	9	9	9	9	9	9	9	9	9
8	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	27	27	27	27
10	9	9	9	9	9	27	27	27	27
11	9	9	9	27	27	27	27	27	27
12	9	9	9	27	27	27	27	27	27
13	9	9	9	27	27	27	27	27	27
14	5	5	9	27	27	27	27	27	27
15	9	9	9	27	27	27	27	27	27
16	5	5	9	27	27	27	27	27	27
17	9	9	27	27	27	27	27	27	63
18	5	5	27	27	27	27	27	27	63
19	9	9	27	27	27	27	27	63	63
20	5	5	27	27	27	27	27	63	63
21	5	5	9	9	9	9	9	9	9
22	9	9	9	9	9	18	18	18	18
23	5	5	9	9	9	9	9	9	9
24	5	5	5	9	9	9	9	9	9
25	9	9	9	9	9	27	27	27	27
26	135	163	279	279	387	459	375	951	951
27	9	9	9	27	27	27	27	63	63
28	5	5	5	9	9	9	9	9	27
29	375	375	375	567	567	567	567	567	567
30	951	951	951	951	951	951	951	951	951
31	549	567	567	951	1335	1335	1335	1335	1719
32	5	5	9	9	9	9	9	18	18
33	18	18	18	18	18	36	36	36	36
34	375	567	567	567	567	567	1335	1719	1719
35	9	9	9	9	9	27	27	27	27
36	23	23	51	99	153	63	63	99	99
37	23	37	51	135	63	63	63	99	99
38	23	37	51	153	207	63	63	99	99
39	23	37	51	153	207	63	63	63	99
40	55	55	115	243	135	135	135	207	279
41	9	9	23	27	27	63	63	63	99
42	27	27	59	63	81	135	135	153	207
43	9	9	9	27	27	27	63	63	99
44	27	27	45	45	45	81	135	153	171
45	15	15	15	15	15	15	15	15	15
46	3	3	3	3	3	3	3	3	3
47	5*	5F	5F	5F	5F	5F	5F	5F	5F
48	49	49	131	187	237	263	289	365	369
49	9	9	9	9	9	9	9	9	9
50	1719 ²	1719 ²	1719 ²	1719 ²	1719 ²	1719 ²	1719 ²	1719 ²	1719 ²

TABLE 4 INTEGRALS IN Appendix A (Absolute Error)

IER=1
F indicates Failure

Case	AOC	10 ⁻¹	10 ⁻²	10 ⁻³	10 ⁻⁴	10 ⁻⁵	10 ⁻⁶	10 ⁻⁷	10 ⁻⁸	10 ⁻⁹
1		78	108	126	126	144	234	252	288	288
2		24	24	24	24	24	24	24	24	24
3		39	545	375	375	375	567	567	567	567
4		131	351	579	2161	4013	6327	6327	6327	6327
5		279	279	459	375	375	375	375	567	567
6		837	873	945	1143	1215	1107	999	1347	1455
7		46	46	90	90	90	126	180	216	288
8		135	279	279	279	279	279	495	375	375
9		95	97	129	163	171	207	297	333	387
10		71	91	97	121	127	153	207	243	297
11		155	263	435	597	759	1007	1263	1523	2087F
12		73	73	91	99	99	99	117	135	135
13		27	69	83	91	167	171	343	401	513
14		21	33	45	69	81	105	133	147	175
15		27	55	83	95	99	171	171	171	297

TABLE 5 INTEGRALS IN APPENDIX A (Relative Error)

† Although IER=2, the accuracy was met.
IER=1 except for superscript values.
F = Failure

CASE	ACC	-10^{-1}	-10^{-2}	-10^{-3}	-10^{-4}	-10^{-5}	-10^{-6}	-10^{-7}	-10^{-8}	-10^{-9}
1		90	90	126	180	180	234	252	252	288
2		24	24	24	24	24	24	24	24	24
3		375	375	375	567	567	567	567	567	567
4		735	689	1731	6327	6327	6327	6327	6327	6327
5		279	279	531	375	375	375	375	567	567
6		891	891 ²⁺	981 ²⁺	1179	1179 ²⁺	1107 ²⁺	999 ²⁺	1419 ²⁺	1767 ²⁺
7		46	46	72	90	90	108	216	216	252
8		135	279	279	279	279	423	375	375	375
9		153	171	207	333	333	387	459	459	567
10		101	105	179	221	239	279	279	387	495
11		135	243	513	729	891	1395	2277	5883	6663 ^F
12		73	73	185	99	99	117	135	135	135
13		27	69	87	99	171	171	351	711	783
14		21	33	57	81	93	119	133	161	189
15		27	55	87	99	99	171	171	279	351

TABLE 6 COMPARISONS OF NL9 VERSUS CADRE AND
NL9 VERSUS FOGIE FOR KAHNER'S TEST INTEGRALS

of cases with less function evaluations

	CADRE	NL9	TIE	FOGIE	NL9	TIE
ACC = 10^{-3}	8	7	6	4	17	0
ACC = 10^{-6}	6	12	3	4	17	0
ACC = 10^{-9}	6	15	0	13	8	0

FAILURES

	FOGIE	CADRE	NL9
ACC = 10^{-3}	0	1	1
ACC = 10^{-6}	0	1	1
ACC = 10^{-9}	0	1	1

APPENDIX A
GROUP 3 TEST INTEGRALS

CASE	INTEGRALS	END POINTS
1	$\frac{1}{1+x^2}$	(-100,+100)
2	$f(x) = \begin{cases} x-5, & 5 < x < 10 \\ 15-x, & 10 < x < 15 \\ 0 & \text{otherwise} \end{cases}$	(0,20)
3	$x \sin (72\pi x)$	(0,1)
4	$\sin^2 (50\pi x) / [50(\pi x)^2]$	(0,10)
5	$J_1(x)$	(0,100)
6	$\sin(\frac{1}{x})/x^2$	(0.004999999995,1)
7	$1/\sqrt{ x^2-1 }$	(-1,1)
8	$4\pi^2 x \sin 20\pi x \cos 2\pi x$	(0,1)
9	$1/[1 + (230x-30)^2]$	(0,1)
10	$1/x^5$	(0.01,1.10)
11	$1/\sqrt{ x-.2 }$	(0,1)
12	$x \log \sin x$	(0,1)
13	$\log (1/x)$	(0,1)
14	$ x-.6 $	(0,1)
15	$x \arccos (\frac{1}{x}) \sqrt{1-(x-1)^2}$	(1,2)

```

SUBROUTINE FINT(XL,XU,ACC,INTVL,LIMIT,Y,ERR,NFUN,IER)
C .....
C
C PURPOSE -
C THIS ROUTINE COMPUTES THE INTEGRAL OF F(X) dx BETWEEN THE LIMITS
C XL AND XU TO WITHIN AN ACCURACY SPECIFIED BY THE USER.
C
C USAGE -
C CALL FINT(XL,XU,ACC,INTVL,LIMIT,Y,ERR,NFUN,IER)
C
C DESCRIPTION OF PARAMETERS -
C FCT - THE DUMMY NAME OF THE SUBROUTINE TO COMPUTE F(X).
C ITS FORM MUST BE
C      SUBROUTINE FCT(X,F)
C      DOUBLE PRECISION X,F,....
C WHERE F = F(X) IS COMPUTED WITHOUT DESTROYING X. FCT
C MUST BE DECLARED EXTERNAL IN THE CALLING PROGRAM.
C XL - THE LOWER LIMIT OF INTEGRATION (A DOUBLE PRECISION
C      PARAMETER).
C XU - THE UPPER LIMIT OF INTEGRATION (A DOUBLE PRECISION
C      PARAMETER).
C ACC - THE ACCURACY DESIRED (A SINGLE PRECISION PARAMETER).
C      IF ACC > 0, ACC IS USED FOR THE ABSOLUTE ERROR.
C      IF ACC < 0, -ACC IS USED FOR THE RELATIVE ERROR.
C INTVL - ABSINTVL IS THE NUMBER OF EQUAL INTERVALS INTO
C        WHICH [XL,XU] IS SUBDIVIDED, WITH THE INTEGRAL OF F(X)
C        FOUND SEPARATELY AND THE RESULTS SUMMED. IF INTVL > 0,
C        THE BASIC SCHEME OF ESTIMATING THE INTEGRAL IS USED,
C        IN WHICH NEAL-LOBATCHEV, 3-, 4-, AND 5-POINT FORMULAS ARE
C        USED, FOLLOWED BY AN OPTIMAL 9-POINT FORMULA USING THE
C        PREVIOUS 5 POINTS. INTERVAL HALVING AND QUEUE STACK-
C        ING OCCURS THEREAFTER. IF INTVL < 0, A 24-POINT GAUSS-
C        LEGENDRE FORMULA IS USED WITH INTERVAL HALVING AND
C        QUEUE STACKING. INTVL = 1 WILL BE MOST EFFICIENT
C        MOST OF THE TIME. INTVL = -1 WILL BE MOST EFFICIENT
C        FOR HIGHLY OSCILLATORY INTEGRANDS.
C LIMIT - THE MAXIMUM NUMBER OF FUNCTION EVALUATIONS ALLOWED
C        BEFORE AJS ABORTS THE PROBLEM.
C Y - THE FINAL ESTIMATE OF THE INTEGRAL, A DOUBLE PRECISION
C      VARIABLE.
C ERR - ABSERRFCT IS THE MAGNITUDE OF THE ESTIMATED ABSOLUTE
C      ERROR. IF ERRFCT > 0, THE RESULT WAS OBTAINED BECAUSE
C      TWO SUCCESSIVE ESTIMATES SATISFIED CERTAIN CRITERIA.
C      IF ERRFCT < 0, THE RESULT WAS OBTAINED BY ACCELERATION.
C      THE SEQUENCE OF ESTIMATES IS REPEATEDLY TRANSFORMED BY
C      ATKIN'S DELTA SQUARED TRANSFORMATION, AND TWO
C      SUCCESSIVE TRANSFORMED VALUES SATISFIED CERTAIN
C      CRITERIA. ERRFCT IS A SINGLE PRECISION VARIABLE.
C NFUN - THE ACTUAL NUMBER OF FUNCTION EVALUATIONS MADE.
C IER - OUTPUT ERROR CODE. IF IER
C      = 1, THE ANSWER IS THOUGHT TO BE WITHIN THE ACCURACY
C        SPECIFIED.
C      = 2, THE ERROR IS THOUGHT TO EXCEED THAT WHICH WAS
C        SPECIFIED BY ACC.
C      = 3, THE RUN WAS ABORTED BECAUSE TO CONTINUE WOULD
C        REQUIRE NFUN GREATER THAN LIMIT.
C      = 4, THE RUN WAS ABORTED BECAUSE THE QUEUE, CURRENT-
C        LY SET TO 16 INTERVALS, WAS FILLED. THE
C        INTEGRAL MAY NOT EXIST, OR THE INTEGRAND IS SO

```

BEST AVAILABLE COPY

C	OSCILLATORY THAT A GREATER MAGNITUDE OF INTVL	00000000
C	IS REQUIRED.	00000010
C		00000020
C	REMARKS -	00000030
C	ALTHOUGH RELIABILITY WAS EXCELLENT FOR A LARGE SET OF TEST	00000040
C	INTEGRALS OF MANY TYPES, THE USER IS CAUTIONED THAT AUTOMATIC	00000050
C	QUADRATURE ROUTINES OF THIS TYPE MUST FAIL STATISTICALLY.	00000060
C		00000070
C	00000080
C	DOUBLE PRECISION XL,XU,Y,XINC,XLL,XUL,A,B,YF	00000090
C	EXTERNAL FCT	00000100
C	COMMON/EVENO/FACT,TAIL	00000110
C	INITIALIZATION.	00000120
C	ERROR=0.	00000130
C	NEUN=0	00000140
C	IER=1	00000150
C	Y=0.0	00000160
C	CHECK	00000170
C	IFAIL	00000180
C	J=2 IF 1-POINT FORMULA IS TO BE USED.	00000190
C	J=1	00000200
C	IF (INTVL.EQ.0) J=2	00000210
C	XU=THE UPPER LIMIT OF A SUBDIVISION.	00000220
C	XU=XL	00000230
C	IN=THE NUMBER OF INTERVALS INTO WHICH (XL,XU) IS SUBDIVIDED.	00000240
C	IN=IABS(INTVL)	00000250
C	ERR=THE MAXIMUM ERROR ALLOWED IN EACH SUBINTERVAL.	00000260
C	ERR=ACC/SQRT(FLDGT(IN))	00000270
C	XINC=THE WIDTH OF EACH SUBDIVISION.	00000280
C	XINC=(XU-XL)/DFLOAT(IN)	00000290
C	TAIL IS A COMMON VARIABLE USED TO CHECK DECAYING INTEGRAND VALUES.	00000300
C	TAIL=0.	00000310
C	MAJOR LOOP.	00000320
C	DO 100 I=1,IN	00000330
C	XLL=THE LOWER LIMIT OF A SUBDIVISION.	00000340
C	XLL=XU	00000350
C	XUL=XLL+XINC	00000360
C	A AND B ARE INTERVAL TRANSFORMATION CONSTANTS.	00000370
C	A=(XU-XLL)*.500	00000380
C	B=(XUL-XLL)*.500	00000390
C	FACT IS THE EVENTUAL CONTROL FOR A SYMMETRIC (EVEN) FUNCTION.	00000400
C	FACT=-1.	00000410
C	5 GOTO(10,25), I	00000420
C	COMPUTE THE ESTIMATE OF THE SUBINTEGRAL.	00000430
C	10 CALL FIRST(FCT,A,B,ERR,Y,YF,NEUN,LIMIT,IER,ERRR)	00000440
C	11 GOTO (100,30,12), IER	00000450
C	NL9 ABORTED. RETURN THE BEST ESTIMATE IF ON THE LAST INTERVAL.	00000460
C	12 IF(I.EQ.IN) GOTO 20	00000470
C	NL9 ABORTED. RETURN Y=0 AND LARGE ERROR.	00000480
C	Y=0.00	00000490
C	15 ERROR=1.E75	00000500
C	GOTO 170	00000510
C	UPDATE THE RESULT Y BEFORE RETURNING.	00000520
C	20 Y=Y+YF	00000530
C	GOTO 15	00000540
C	FOR J=2, USE THE 96-POINT FORMULA.	00000550
C	25 CALL Q96FCT,A,B,YF,NEUN,LIMIT,IER)	00000560
C	GOTO 11	00000570
C	GO TO THE INTERVAL BISECTING AND CUBE STACKING ROUTINE.	00000580

BEST AVAILABLE COPY

30	CALL GSUBROUTINE FCT, X0, X1, Y0, Y1, NEON, LIMIT, IER, ERROR	00001150
	IF (IER.GT.2) GOTO 12	00001200
C	UPDATE Y AND CONTINUE.	00001210
100	Y=Y+YF	00001220
	IER=1	00001230
C	TERMINATION PHASE. SET IER AND RETURN.	00001240
	IF (ERROR) 130,120,110	00001250
110	ERROR=SQR(ABS(ERROR))	00001260
	GOTO 140	00001270
C	IF ERROR=0, SET ERROR TO 1 PART IN 10% IS.	00001280
120	ERROR=1.E-15*ABS(Y)	00001290
	GOTO 140	00001300
130	ERROR=-SQRT(-ERROR)	00001310
140	ERR=ACC	00001320
C	ADJUST TEST FOR RELATIVE ERROR REQUEST.	00001330
	IF (ERR.LT.0.1) ERR=ACC*ABS(Y)	00001340
	IF (ABS(ERROR).LE.ERR) GOTO 150	00001350
C	THE TEST FAILS. THE RESULT MAY EXCEED THE ACCURACY SPECIFIED.	00001360
	IER=2	00001370
	GOTO 170	00001380
C	THE TEST PASSES. THE RESULT IS WITHIN THE ACCURACY SPECIFIED.	00001390
150	IF (ABS(ERROR).LT.ERR*20.) GOTO 160	00001400
	ERROR=SIGN(ABS(ACC),ERROR)	00001410
	IF (ACC.LT.0.) ERROR=ERROR*ACOS(1)	00001420
	GOTO 170	00001430
C	ADJUST THE EMPIRICAL CONSTANT.	00001440
160	ERROR=ERROR*20.	00001450
170	RETURN	00001460
	END	00001470
C	00001480
	SUBROUTINE FCT(FCT,A,B,F,Y,YE,NEON,LIMIT,IER,ERROR)	00001490
C	PURPOSE -	00001500
C	THIS ROUTINE IS CALLED TO OBTAIN THE ESTIMATE OF THE INTEGRAL	00001510
C	OF A SUBINTERVAL USING THE BASIC SQUARE REQUIREMENT OF TO NINE	00001520
C	FUNCTION EVALUATIONS.	00001530
C		00001540
C	DESCRIPTION OF PARAMETERS -	00001550
C	FCT - SEE NL9.	00001560
C	A,B - TRANSFORMATION PARAMETERS.	00001570
C	ERR - ERROR REQUIREMENT FOR THIS INTERVAL.	00001580
C	Y - THE VALUE OF THE TOTAL INTEGRAL SO FAR.	00001590
C	YE - THE RETURNED ESTIMATE OF THE INTEGRAL.	00001600
C	NEON - SEE NL9.	00001610
C	LIMIT - SEE NL9.	00001620
C	ERROR - ESTIMATE OF THE SUM OF SQUARES OF ERROR SO FAR.	00001630
C	IER = 1 IF THE INTEGRAL WAS FOUND SATISFACTORILY.	00001640
C	= 2 IF THE ERROR CRITERIA WAS NOT MET.	00001650
C	= 3 IF THE INTEGRAL IS TO BE ABORTED.	00001660
C	00001670
	DOUBLE PRECISION A,B,Y,YE,F(3),Y(3),Y(3)	00001680
	COMMON/VELOC/FACI,TALE	00001690
	EXTERNAL FCT	00001700
C	K CONTROLS PARAMETERS IN SUBROUTINE REMOVE.	00001710
	K=1	00001720
C	I CONTROLS PATH AFTER RETURN FROM REMOVE.	00001730
	I=1	00001740
C	FIND 3-POINT ESTIMATE.	00001750
	CALL GSUBROUTINE FCT,A,B,F,YE,NEON,LIMIT,IER	00001760
C	INITIALIZE OF THE FIRST TIME THROUGH. OF CONTROLS BYPASSING	00001770

BEST AVAILABLE COPY

C	REMOVE IF THERE IS UNDERFLOW.	00001730
	IF (NFUN.GT.3) GOTO 2	00001740
	UF=1.	00001750
2	IF (UF.EQ.0.1) GOTO 3	00001760
	CALL OVERFL(1)	00001770
C	DO NOT BYPASS REMOVE IF 1- UNDERFLOW OR 3-POINT ESTIMATE IS NOT 0.	00001780
	IF (YEST(2).NE.C.00.0R.J.NE.3) UF=0.	00001790
C	CHECK FOR ABORTING INTEGRAL.	00001800
3	IF (IER.EQ.2) GOTO 5	00001810
	YE=0.00	00001820
4	RETURN	00001830
5	IF (UF.EQ.1.1) GOTO 8	00001840
C	CHECK ERROR CRITERIA IN REMOVE.	00001850
	CALL REMOVE(YEST,Y,YE,IER,K,CRR,R,IER)	00001860
	IF (IER.EQ.1) GOTO 4	00001870
C	CONTINUE WITH BASIC SCHEME.	00001880
8	GOTO (10,15,4), I	00001890
C	TURN OFF TAIL INDICATOR AFTER 3 FUNCTION EVALUATIONS.	00001900
10	IF (FOR.EQ.3.AND.TAIL.EQ.1.E75) TAIL=0.	00001910
C	FIND 3-POINT ESTIMATE OF THE INTEGRAL.	00001920
	CALL G35(FCT,A,B,F,YEST(1),NFUN,LIMIT,IER)	00001930
C	CHECK FOR ABORTING INTEGRAL.	00001940
	IF (IER.EQ.2) GOTO 4	00001950
	K=2	00001960
	I=2	00001970
	GOTO 5	00001980
C	FIND 9-POINT ESTIMATE OF THE INTEGRAL.	00001990
15	CALL G95(FCT,A,B,F,YEST(1),NFUN,LIMIT,IER)	00002000
C	CHECK FOR ABORTING INTEGRAL.	00002010
	IF (IER.EQ.2) GOTO 4	00002020
	I=3	00002030
	K=2	00002040
	GOTO 5	00002050
	END	00002060
C	00002070
	SUBROUTINE G13(FCT,A,B,F,Y,NFUN,LIMIT,IER)	00002080
C	PURPOSE -	00002090
C	THIS ROUTINE IS CALLED TO OBTAIN A 1- AND 3-POINT ESTIMATE OF	00002100
C	THE INTEGRAL OF THE SUBINTERVAL.	00002110
C	DESCRIPTION OF PARAMETERS -	00002120
C	FCT,A,B,NFUN,LIMIT - SEE SUBROUTINE FIRST.	00002130
C	F(2) = SUM OF VALUES OF THE FUNCTION AT THE OUTER POINTS.	00002140
C	F(3) = FUNCTION VALUE @ THE MIDPOINT.	00002150
C	Y(1) = 1-POINT INTEGRAL ESTIMATE.	00002160
C	Y(2) = 3-POINT INTEGRAL ESTIMATE.	00002170
C	IER = 2 IF COMPUTATION IS CARRIED OUT.	00002180
C	= 3 IF LIMIT WOULD BE EXCEEDED (INTEGRAL TO BE ABORTED).	00002190
C	00002200
	DOUBLE PRECISION A,B,C,F(3),Y(4),X1,X2,X3	00002210
	COMMON/VELOC/FAC,TAIL	00002220
	DATA X1,X2,X3/240FFF9724143BFF,Z405559B5F6711903,	00002230
	* Z411554C9809100067	00002240
	NFUN=NFUN+3	00002250
C	CHECK FOR ABORTING INTEGRAL DUE TO LIMIT LIMITATIONS.	00002260
	IF (NFUN.LE.LIMIT) GOTO 10	00002270
	NFUN=NFUN-3	00002280
	IER=3	00002290
	GOTO 20	00002300

BEST AVAILABLE COPY

```

10 IER=2
C FIND FUNCTION AT MIDPOINT.
  CALL FCTB,F(3)
  Y(1)=F(3)*2.DC*A
C FIND FUNCTION AT OUTER POINTS.
  C=A*X1
  CALL FCTB-C,F(1)
  CALL FCTB+C,C)
C SET TAIL IF VALUES ARE OF OPPOSITE SIGN.
  IF(F(3)*F(1).LT.0.DD) IF (F(3)*C.LT.0.DD) TAIL=1.E75
C CHECK FOR SYMMETRIC FUNCTION.
  IF(FACT.LT.0.) CALL EVEN(C,F(1))
C STORE SUM OF OUTER FUNCTION VALUES IN F(2).
  F(2)=C+F(1)
C COMPUTE 3-POINT ESTIMATE OF THE INTEGRAL.
  Y(2)=A*(F(2)*K2+F(3)*K3)
  IF(FACT.NE.2.) GOTO 20
C DOUBLE VALUES FOR SYMMETRIC FUNCTION.
  Y(1)=Y(1)*2.DC
  Y(2)=Y(2)*2.DC
20 RETURN
  ENF
C .....
SUBROUTINE G35(FCT,A,B,C,F,Y,NFUN,LIMIT,IER)
C PURPOSE -
C THIS ROUTINE OBTAINS A 5-POINT ESTIMATE OF THE INTEGRAL USING
C THE 3 FUNCTION VALUES FOUND IN SUBROUTINE G13.
C DESCRIPTION OF PARAMETERS -
C FCT,A,B,NFUN,LIMIT,IER - SEE SUBROUTINE G13.
C F(2),F(3) - SEE SUBROUTINE G13.
C F(1) - SUM OF FUNCTION VALUES BETWEEN THE OUTER POINTS AND
C THE MIDPOINT.
C Y - ESTIMATE OF THE INTEGRAL OF THIS SUBINTERVAL.
C .....
  DOUBLE PRECISION A,B,C,F(3),Y,D,X1,X2,X3,X4
  COMMON/EVEN/FACT,TAIL
  DATA X1,X2,X3,X4/Z40199E040FF57F5,Z40479317E9A16158,
  * Z406650E2+29555784,Z406606B71158A100/
  NFUN=NFUN+2
C CHECK FOR ABORTING DUE TO LIMIT LIMITATIONS.
  IF(NFUN.LE.LIMIT) GOTO 10
  NFUN=NFUN-2
  IER=3
  GOTO 20
10 IER=2
C CONTRIBUTION FROM OUTER POINTS.
  Y=X1*F(2)
C COMPUTE CONTRIBUTION FROM NEW POINTS.
  C=A*X2
  CALL FCTB-C,D)
  CALL FCTB+C,C)
C CHECK FOR SYMMETRIC FUNCTION.
  IF(FACT.LT.0.) CALL EVEN(C,D)
C STORE SUM OF NEW FUNCTION VALUES IN F(1).
  F(1)=C+D
C SUM WITH CONTRIBUTION OF REMAINING FUNCTION VALUES.
  Y=A*(Y+X3*(C+D)+X4*F(3))
C DOUBLE VALUE IF SYMMETRIC FUNCTION.

```

```

      IF (FACT.EQ.2.) Y=Y*Y
20 RETURN
      END
C-----
      SUBROUTINE G59FACT(A,B,C,Y,FUN,LIMIT,ITER)
      PURPOSE -
      C THIS ROUTINE CORRECTS THE Y-POINT ESTIMATE OF THE INTEGRAL IN
      C THIS SUBINTERVAL USING THE 5 FUNCTION VALUES PREVIOUSLY FOUND.
      C
      C DESCRIPTION OF PARAMETERS -
      C IDENTICAL TO SUBROUTINE G39.
C-----
      DOUBLE PRECISION A,X(6),F(5),Y,X1,X2,X3,X4,X5,X6,X7
      COMMON/EVERC/FACT,TAIL
      DATA X1,X2,X3,X4,X5,X6,X7/74.0E-3,3.0E-3,1.0E-3,1.0E-3,1.0E-3,1.0E-3,1.0E-3/
      * 24097489165EFC7E4,23E70EAL75E22C2C,24020E2E6E120E22C,
      * 24048E0E763E38E3,24025E1E2635E38E3,24057E6E6E0EFCF17/
      NEUN=NEUN+4
      C CHECK FOR ADDING DUE TO LIMIT LIMITATIONS.
      IF (FUN.LE.LIMIT) GOTO 10
      NEUN=NEUN-4
      ITER=5
      GOTO 20
10 ITER=2
      C CONTRIBUTION FROM OUTLIER POINTS.
      Y=X*F(2)
      C=A*X1
      CALL FCTH(C,1)
      CALL FCTH(C,1)
      C CHECK FOR SYMMETRIC FUNCTION.
      IF (FACT.EQ.2.) CALL EVENC(0)
      C ADD NEW POINTS AND CONTRIBUTION FROM OLD INTERMEDIATE POINTS.
      Y=Y+(D+C)*X4+(E1)*X5
      C=A*X2
      CALL FCTH(C,1)
      CALL FCTH(C,1)
      C CHECK FOR SYMMETRIC FUNCTION.
      IF (FACT.EQ.2.) CALL EVENC(0)
      C ADD NEW POINTS AND CONTRIBUTION FROM MIDPOINT.
      Y=A*(Y+X5*(C*2)+F(5)*X7)
      C DOUBLE VALUE FOR SYMMETRIC FUNCTION.
      IF (FACT.EQ.2.) Y=Y*Y
20 RETURN
      END
C-----
      SUBROUTINE EVENC(A)
      PURPOSE -
      C THIS ROUTINE SETS FACT = 1, IF THERE IS NO POSSIBILITY OF
      C HAVING A SYMMETRIC FUNCTION.
      C
      C DESCRIPTION OF PARAMETERS -
      C A,B = FUNCTION VALUES OF POINTS EQUIDISTANT FROM THE MIDPOINT.
C-----
      COMMON/EVERC/FACT,TAIL
      DOUBLE PRECISION A,B,S
      C IF A=B, FUNCTION MAY BE SYMMETRIC.
      IF (A.EQ.B) GOTO 10
      S=A+B
      C IF A=-B, THE FUNCTION IS ASSUMED NOT SYMMETRIC.

```

BEST AVAILABLE COPY

```

      IF (ABS(1.0-0.01) < 0.001) GOTO 5
      2 FACT=1.
      GOTO 10
      C IF ABS((A-B)/(A+B)) > 1.0-11, THE FUNCTION IS ASSUMED NOT
      C SYMMETRIC.
      5 IF (CAPS((A-B)/S).GT.1.0-11) GOTO 3
      10 RETURN
      END
C .....
SUBROUTINE REMOVE(YEST,Y,YE,ERR,K,LEFT)
C PURPOSE -
C THIS ROUTINE DECIDES WHETHER THE INTEGRAL OF THE SUBINTERVAL
C OR ITS RIGHT- AND LEFT-HAND PART ARE SUFFICIENTLY ACCURATE FOR
C REMOVAL FROM THE QUEUE.
C
C DESCRIPTION OF PARAMETERS -
C YEST(1) = FIRST ESTIMATE OF THE INTEGRAL.
C YEST(2) = SECOND ESTIMATE OF THE INTEGRAL USING MORE POINTS.
C YEST(3),YEST(4) = LEFT- AND RIGHT-HAND ESTIMATES OF THE
C INTEGRAL IF REMOVE IS CALLED FROM SUBROUTINE INPAF.
C Y = SUM OF INTEGRALS OF SUBINTERVALS SUCCESSFULLY
C COMPUTED SO FAR.
C YE = BEST ESTIMATE OF THE INTEGRAL OF THIS INTERVAL SO FAR.
C ERROR,ERR = SEE SUBROUTINE FIRST.
C K = 1 IF THE TEST COMPARES A 1- AND 3-POINT ESTIMATES.
C     = 2 IF THE TEST COMPARES A 3- AND 5-POINT OR A 5- AND
C     AND 9-POINT ESTIMATES.
C     = 3 IF THE TEST COMPARES RESULTS BEFORE AND AFTER
C     BISECTION.
C     = 4 IF THE COMPARISON USES A 95-POINT FORMULA. IN THIS
C CASE, K=1 AS OUTPUT MEANS THE RIGHT-HAND PART OF THE
C SUBINTERVAL SHOULD BE REMOVED FROM THE QUEUE.
C IF K=1, FOR REMOVAL OF THE INTERVAL FROM THE QUEUE, IF IF
C K=4 ON INPUT, FOR REMOVAL OF THE LEFT-HAND PART OF
C THE SUBINTERVAL.
C     = 2, FOR NO REMOVAL OF THE INTERVAL FROM THE QUEUE, OR
C IF K=4 ON INPUT, FOR NO REMOVAL OF THE LEFT-HAND
C PART FROM THE QUEUE, AFTER BISECTION.
C .....
      DOUBLE PRECISION YEST(4),Y,YE,YA
      COMMON ZEVLMC/FACT,IT,IF
      C IT IS THE TEST NUMBER.
      IT=1
      IF K=2
      C YA AND YE ARE TESTING QUANTITIES.
      YA=PARSYEST(2)
      YE=PARSYEST(3) - YEST(1)
      C EK IS A FACTOR IN THE TEST. TEST 1 REQUIRES THE 2 ESTIMATES
      C YEST(1) AND YEST(2) TO AGREE TO ABOUT 15 DIGITS.
      EK=10.
      ETEST=1.0-14*YA
      C BASIC TEST.
      10 IF (YE.GT.ETEST/EK) GOTO (15,10,40,50), K
      C TEST PASSES. ADJUST ERROR.
      IF (YE.LT.YA*.0-14) EK=EK*10.
      ERR=EK*YLC*.100
      ERROR=SIGN(ABS(ERROR) + ERROR,ERROR)
      IF (IT.NE.4) IF K=1
      C TEST RIGHT-HAND INTERVAL IF K=4 AND IT=3.

```

BEST AVAILABLE COPY

	IF (EK, 4) GOTO (50, 50, 72, 50), IT	00004140
C	PREPARE TO RETURN, UPDATE YF AND YEST(1).	00004150
14	YE=YEST(2)	00004160
	YEST(1)=YEST(2)	00004170
	RETURN	00004180
C	SECOND TEST FOR K=1 IF IT=1.	00004190
15	IF (IT, EK, 2) GOTO 14	00004200
C	ADJUST TAIL VALUE IF YA WAS SMALL. MT=0 MEANS THIS WAS DONE.	00004210
	MT=0	00004220
	IF (ABS(YA)*8, 00, 06, ABS(TAIL)) GOTO 21	00004230
	MT=1	00004240
	GOTO 16	00004250
21	TAIL=YA	00004260
C	SET UP FOR TEST 2.	00004270
16	EK=1, E2	00004280
	IT=2	00004290
17	YE=YA	00004300
C	BYPASS VALUE TEST IF TAIL WAS SMALL.	00004310
	IF (MT, 00, 0) GOTO 14	00004320
C	SET UP TO TEST VALUE OF ESTIMATE WITH ERROR REQUIREMENTS USING	00004330
C	VALUE OF INTEGRAL SE FAN.	00004340
23	FTEST=ABS(EPF)	00004350
	IF (ERR, LT, 0, 0) FTEST=FTEST*QUANTITY/10, 10	00004360
	GOTO 10	00004370
C	CONTROL WHEN K=2.	00004380
18	GOTO (19, 35, 14), IT	00004390
C	SECOND TEST.	00004400
19	IT=2	00004410
C	PREPARE TO TEST ERROR.	00004420
20	FTEST=ABS(EPF)	00004430
	IF (ERR, LT, 0, 0) FTEST=FTEST*YA	00004440
C	BYPASS TEST IF RELATIVE ERROR IS TOO LARGE AND THERE CAN BE A TAIL	00004450
	IF (YE, GE, 100*YA, AND, TAIL, NE, 1, E75) GOTO (19, 10, 40, 50), K	00004460
C	ADJUST EK DEPENDING ON THE RELATIVE ERROR.	00004470
	IF (YE, LT, 1, 1-3*YA) GOTO 22	00004480
	EK=100.	00004490
	GOTO 10	00004500
22	IF (YE, LT, 1, 1-5*YA) GOTO 24	00004510
	EK=10.	00004520
	GOTO 10	00004530
24	EK=1.	00004540
	GOTO 10	00004550
C	K=2 AND IT=2.	00004560
35	EK=1, E2	00004570
	IT=3	00004580
	GOTO 17	00004590
C	CONTROL FOR K=3.	00004600
40	GOTO (19, 14), IT	00004610
C	INDICATE REMOVAL OF RIGHT-HAND INTERVAL.	00004620
50	K=1	00004630
	GOTO 14	00004640
C	CONTROL FOR K=4.	00004650
60	GOTO (19, 75, 72, 14), IT	00004660
C	TEST RIGHT-HAND INTERVAL.	00004670
72	IT=4	00004680
	EK=10.	00004690
	YE=ABS(YEST(4))	00004700
	GOTO 10	00004710
C	TEST LEFT-HAND INTERVAL.	00004720

BEST AVAILABLE COPY


```

75  I=3
   EX=10.
   YE=ABS(YE-10)
   GOTO 23
   END
C .....
SUBROUTINE W9C(FCT,A,B,Y,X,FUN,LIMIT,IER)
C PURPOSE -
C THIS ROUTINE COMPUTES A 90-POINT ESTIMATE OF THE INTEGRAL IN
C THIS SUBINTERVAL.
C
C DESCRIPTION OF PARAMETERS -
C IDENTICAL TO SUBROUTINE G59.
C DOUBLE PRECISION A,B,Y,X(48),W(48),F1,F2
C COMMON/WEVENT/FACT,TALE
C DATA X/
* 240FEFA6B05536A2,Z40FE94CE07D2E5AF,Z40FEF8A4F30A24FA,
* 240FE175867BC1ECL,Z40FECE1107E65TAGE,Z40FEF86435AC23105,
* 240FE9D72655BA7B20,Z40FE7E4449C1600AF,Z40FE5AE21C220C00E,
* 240FE335581BC35FCH,Z40FE3745316120BCA,Z40FE7E90456SL14C,
* 240FA421EFCCEFA1,Z40F0CC1FC8501EB3,Z40FE3CH86E33EFC7,
* 240DF13512EFA0A9C,Z40DAE1E8C0057365,Z40CCE7E5315F3EC,
* 240D1C4380CF35F7C,Z40CCE1C8AA07C803,Z40C7C5440238AFAB,
* 240C2754C805193A1,Z40DCE1958653F57C,Z40B7369877E5A3D,
* 240B148E8C33F5031,Z40BCE11F83F3031,Z4044E7E453675653,
* 2409E71E081CF877,Z4057C0F6193J7058,Z40910CF3A89C5145,
* 2408A1556C01AB34F,Z40B2FE658C9E045C,Z4078C316CFAE8C,
* 24074C7D631C4E34F,Z40CCEC30E4583265,Z406552FA7C440CE,
* 2405C5E4F3C45FF7,Z40550C390B240F24,Z404E1A0C6A0F24F8,
* 24045F05B7062745C,Z403CE2E125C16958,Z4035C3C899C6C67,
* 2402D57C1D045C0F3,Z4025FE8E8A34671C,Z401E12285FA8EEF,
* 24014CCE9A444333,Z3FC7F31FEC765321,Z3F42A867693764937
DATA W/
* Z3E3437F30CFE7CFA,Z3E740C4CF45340E1,Z3E8EC1F9F7253456,
* Z3F1J3D22F3A11D93,Z3F14E5C6CCE7077E,Z3F18C3066C51445,
* Z3F1C1105F70FE24,Z3F214945E904F60C,Z3F25751367C233C,
* Z3F299E5712667004,Z3F2D0E4FF745201B,Z3F31C5E16A7FAAF,
* Z3F35C5F24A4FCA77,Z3F39F70C014F9543,Z3F319930EFC0F73F,
* Z3F41E4595507EC93,Z3F4529857C650A92,Z3F48064E1F77206F,
* Z3F4C6F2406884A7L,Z3F4FF3307251734,Z3F5261A52C777B55,
* Z3F5669CFCC5AAAA5,Z3F59F9C23773C036,Z3F5E213FA75F202E,
* Z3F603020F12AE1FH,Z3F6324508E72FFCF,Z3F65FF341571F295,
* Z3F68FC204668E58D,Z3F6B50A1A35192C2,Z3F6EC2023355035,
* Z3F7J4B932050F29E,Z3F7240A061B05413,Z3F74F7355E521C4,
* Z3F76C3CE789C5FE0,Z3F787C3110FF5145,Z3F7A749C59E2CA2,
* Z3F7C18C39C014452,Z3F7D414405F0083,Z3F7E94804B0A26,
* Z3F8045A6FC4218AA,Z3F81E3D0269C13A36,Z3F829FDF06E15C05,
* Z3F833C8F00C8EA9F,Z3F83EAA745192455,Z3F847AF4C28CF7F5,
* Z3F84E7512FFF7153,Z3F852CF4F2730328C,Z3F8555C8C85076CZ
NFUN=NFUN*90
C CHECK FOR ABORTING DUE TO LIMIT LIMITATIONS.
IF NFUN.LE.LIMIT GOTO10
NFUN=NFUN-90
IER=3
GOTO 20
10 Y=0.00
IER=2
DO 5 I=1,48
F2=A*X(I)

```

BEST AVAILABLE COPY

```

      CALL FCT(B+F2,F1)
      CALL FCT(B-F2,F2)
C     CHECK FOR SYMMETRIC FUNCTION.
      IF(FACT.LT.0.) CALL EVENIF1,F2)
5     Y=Y+(F1+F2)*W11)
      Y=Y*A
C     DOUBLE VALUE FOR SYMMETRIC FUNCTION.
      IF(FACT.EQ.2.) Y=Y+Y
20    RETURN
      END
C.....
SUBROUTINE INHAF(FCT,XL,XU,J,ERR,Y,YE,NFUN,LIMIT,IER,ERRCR)
C     PURPOSE -
C     THIS ROUTINE DOES THE INTERVAL BISECTING FROM AN INPUT INTERVAL
C     AND CREATES THE QUEUE OF INTERVALS TO BE PROCESSED.
C
C     DESCRIPTION OF PARAMETERS -
C     FCT,ERR,NFUN,LIMIT,ERRCR - SEE SUBROUTINE FIRST.
C     XL = LOWER LIMIT ON THIS SUBINTERVAL.
C     XU = UPPER LIMIT ON THIS SUBINTERVAL.
C     J = 2 FOR A 96-POINT FORMULA.
C     = 1 FOR THE BASIC SCHEME.
C     Y = VALUE OF TOTAL INTEGRAL'SL FAW.
C     YE = THE ESTIMATE OF THE INTEGRAL ON INPUT. ON OUTPUT, IT IS
C     UPDATED TO THE FINAL VALUE.
C     IER = IDENTICAL MEANING AS IN SUBROUTINE ALY.
C.....
      DOUBLE PRECISION XL,XU,Y,YE,A,B,Z(20),FILE(16,3),LAREA,RAREA,MID,
      LPT,PPT,SUNY,UMQ,E,YEST(4)
      INTEGER POINT,CYCLE,FILLND
      COMMON/EVENO/FACT,TAIL
      EXTERNAL FCT
      LEN = LENGTH OF THE QUEUE.
      LEN=16
C     Z = ARRAY USED FOR ACCELERATING THIS INTERVAL TO THE LIMIT.
      Z(1)=YE
      IF(YE.EQ.0.EQ) FACT=1.
C     FILE = ARRAYS OF LOWER AND UPPER INTERVALS, AND THE INTEGRAL
C     ESTIMATE.
      FILE(1,1)=XL
      FILE(1,2)=XU
      FILE(1,3)=YE
C     NFILE = THE NUMBER OF INTERVALS IN THE QUEUE.
      NFILE=1
C     FILLND = POINTER SPECIFYING WHICH INTERVAL IS BEING PROCESSED.
      FILLND=1
C     ISW = 2 ONLY IF J IS SET FROM 1 TO 2 IN THIS ROUTINE.
      ISW=1
C     ISW1 = 2 WILL BE SET TO INDICATE A SYMMETRIC FUNCTION FOR THE
C     FIRST INTERVAL.
      ISW1=1
C     SET UP SYMMETRIC FUNCTION INDICATOR.
      IF(FACT.GT.0.) GOTO 3
      ISW1=2
      FACT=2.
C     ADJUST NEW LOWER LIMIT FOR SYMMETRIC FUNCTION.
      FILE(1,1)=(XL+XU)*.5DC
C     POINT = NEXT WRITING INDEX FOR FILE.
      3 POINT=2

```

BEST AVAILABLE COPY

```

C      SUMY ACCUMULATES THE INTEGRAL ESTIMATES FOR INTERVALS REMOVED FROM THE QUEUE. 00005910
C      SUMY=0.00 00005920
C      IZ IS THE INDEX FOR Z. 00005930
C      4 IZ=1 00005940
C      BEGIN NEW CYCLE. CYCLE MARKS THE END OF A SET OF INTERVALS. 00005950
C      5 CYCLE = POINT 00005960
C      SUMQ ACCUMULATES THE INTEGRAL ESTIMATES FOR INTERVALS PLACED BACK ON THE QUEUE. 00005970
C      SUMQ = 0.00 00005980
C      EVALUATE NEW INTERVAL ON THE STACK. FIRST CHECK IF QUEUE IS FILLED. 00005990
C      10 IF(NFILE.EQ.LEN) GOTO 40 00006000
C      LPT = LEFT END POINT, RPT = RIGHT END POINT, E = ESTIMATE OF 00006010
C      INTEGRAL BETWEEN LPT AND RPT. 00006020
C      12 LPT = FILE(FILEND,1) 00006030
C      RPT = FILE(FILEND,2) 00006040
C      E = FILE(FILEND,3) 00006050
C      UPDATE INTERVALS ON THE QUEUE. 00006060
C      NFILE=NFILE-1 00006070
C      UPDATE FILEND POINTER. 00006080
C      IF(FILEND.EQ.LEN) FILEND=0 00006090
C      FILEND=FILEND+1 00006100
C      CHECK FOR ADJUSTMENT DUE TO SYMMETRIC FUNCTION THE FIRST TIME 00006110
C      THROUGH. 00006120
C      IF(ISW1.EQ.1) GOTO 15 00006130
C      MID=RPT 00006140
C      RAREA=0.00 00006150
C      GOTO 17 00006160
C      15 MID=(LPT+RPT)*.500 00006170
C      COMPUTE THE TRANSFORMATION PARAMETERS FOR THE LEFT INTERVAL. 00006180
C      17 A=(MID-LPT)*.500 00006190
C      B=(MID+LPT)*.500 00006200
C      GOTO (20,75), J 00006210
C      20 K1=1 00006220
C      CALL FIRST(FCT,A,B,ERR,Y+Z(IZ),LAREA,NFUN,LIMIT,IEF,ERROR) 00006230
C      GOTO (30,33,25), IER 00006240
C      25 YE=Z(IZ) 00006250
C      RESET J BEFORE RETURNING. 00006260
C      IF(ISW1.EQ.2) J=1 00006270
C      GOTO 73 00006280
C      K1 = 2 MEANS THE LEFT INTERVAL WILL BE REMOVED FROM THE QUEUE. 00006290
C      30 K1=2 00006300
C      33 GOTO(35,40), ISW1 00006310
C      COMPUTE THE TRANSFORMATION PARAMETERS FOR THE RIGHT INTERVAL. 00006320
C      35 A=(RPT-MID)*.500 00006330
C      B=(RPT+MID)*.500 00006340
C      GOTO (37,40), J 00006350
C      37 K2=1 00006360
C      CALL FIRST(FCT,A,B,ERR,Y+Z(IZ),RAREA,NFUN,LIMIT,IEF,ERROR) 00006370
C      GOTO(40,45,25), IER 00006380
C      K2 = 2 MEANS THE RIGHT INTERVAL WILL BE REMOVED FROM THE QUEUE. 00006390
C      40 K2=2 00006400
C      IF(K1.EQ.2) GOTO 50 00006410
C      TEST FOR REMOVAL OF BOTH RIGHT AND LEFT INTERVALS DUE TO THE SUM 00006420
C      SATISFYING CONDITIONS SPECIFIED IN SUBROUTINE REMOVE. 00006430
C      45 K=3 00006440
C      YEST(1)=E 00006450
C      YEST(2)=LAREA+RAREA 00006460
C      CALL REMOVE(YEST,Y+Z(IZ),YE,ERROR,ERR,K,IER) 00006470
C      00006480
C      00006490

```

BEST AVAILABLE COPY

	IF(IER.EQ.2) GOTO 50	00006500
	K1=2	00006510
	K2=2	00006520
C	CHECK FOR LEFT INTERVAL REMOVAL.	00006530
	50 IF(K1.EQ.1) GOTO 55	00006540
	52 SUMY=SUMY+LAREA	00006550
	GOTO 60	00006560
C	PUT LEFT INTERVAL BACK ON THE QUEUE.	00006570
	55 SUMQ=SUMQ+LAREA	00006580
	FILE(POINT,1)=LPT	00006590
	FILE(POINT,2)=MID	00006600
	FILE(POINT,3)=LAREA	00006610
	IF(POINT.EQ.LEN) POINT=0	00006620
	POINT=POINT+1	00006630
	NFILE=NFILE+1	00006640
C	CHECK FOR RIGHT INTERVAL REMOVAL.	00006650
	60 IF(K2.EQ.1) GOTO 65	00006660
	SUMY=SUMY+RAREA	00006670
	GOTO 70	00006680
C	PUT RIGHT INTERVAL BACK ON THE QUEUE.	00006690
	65 IF(NFILE.EQ.LEN) GOTO 92	00006700
	FILE(POINT,1)=MID	00006710
	FILE(POINT,2)=RPT	00006720
	FILE(POINT,3)=RAREA	00006730
	IF(POINT.EQ.LEN) POINT=0	00006740
	POINT=POINT+1	00006750
	NFILE=NFILE+1	00006760
	SUMQ=SUMQ+RAREA	00006770
	70 IF(NFILE.GT.0) GOTO 85	00006780
C	QUEUE IS EMPTY.	00006790
	YE=SUMY	00006800
	73 RETURN	00006810
C	COMPUTE ESTIMATE WITH 96-POINT FORMULA.	00006820
	75 K1=1	00006830
	K2=1	00006840
	CALL Q96(FCT,A,B,LAREA,NFUN,LIMIT,IER)	00006850
	IF(IER.EQ.2) GOTO (35,82),ISWL	00006860
	GOTO 25	00006870
	80 CALL Q96(FCT,A,B,RAREA,NFUN,LIMIT,IER)	00006880
	IF(IER.EQ.3) GOTO 25	00006890
C	TEST FOR REMOVAL OF INTERVAL WHEN COMPUTED WITH 96-POINT FORMULA.	00006900
	82 K=4	00006910
	YEST(1)=E	00006920
	YEST(2)=LAREA+RAREA	00006930
	YEST(3)=LAREA	00006940
	YEST(4)=RAREA	00006950
	CALL REMOVE(YEST,Y+Z(I,Z),YE,EPRCR,ERR,K,IER)	00006960
	IF(IER.EQ.1) K1=2	00006970
	IF(K.EQ.1) K2=2	00006980
	GOTO 50	00006990
C	CHECK FOR END OF CYCLE.	00007000
	85 IF(FILEND.NE.CYCLE) GOTO 10	00007010
C	CYCLE IS ENDED. BEGIN ACCELERATION PROCEDURES.	00007020
	IZ=IZ+1	00007030
C	RECORD NEW ESTIMATE AFTER BISECTIONS.	00007040
	Z(I,Z)=SUMY+SUMQ	00007050
	CALL ACCEL(IZ,EPR,ERRCR,J,IZ,IER)	00007060
	IF(ISWL.EQ.1) GOTO 85	00007070
C	RESET ISWL TO 1 TO SHOW IT IS NOT THE FIRST TIME THROUGH A CYCLE.	00007080

BEST AVAILABLE COPY

	ISW1=1	00007107
	IF (FACT.EQ.2.) GOTO 83	00007108
C	THE FIRST TIME THROUGH WITH J = 2. START NEW Z ARRAY.	00007109
	Z(1)=Z(12)	00007110
	I2=1	00007111
	88 GOTO (25,5), IFR	00007112
C	QUEUE IS FILLED. TRY THE MORE INTERVAL IF J = 2.	00007113
	90 IF (J.NE.2) GOTO 95	00007114
	GOTO 12	00007115
	92 IER=4	00007116
	GOTO 25	00007117
C	SWITCH TO J = 2 (90-POINT FORMULAT. COMBINE CONTIGUOUS INTERVALS	00007118
C	ON THE QUEUE.	00007119
	95 J=2	00007120
	ISW=2	00007121
	POINT=CYCLE	00007122
	NFILE=1	00007123
	I=0	00007124
	96 SUMQ=FILE(CYCLE,3)	00007125
	FILE(POINT,1)=FILE(CYCLE,1)	00007126
	97 I=I+1	00007127
	IF (I.EQ.LEN) GOTO 98	00007128
	K=CYCLE+1	00007129
	IF (K.GT.LEN) K=1	00007130
	IF (FILE(CYCLE,2).EQ.FILE(K,1)) GOTO 95	00007131
C	CHECK EQUALITY OF ENDPOINTS OF ADJACENT INTERVALS.	00007132
	NFILE=NFILE+1	00007133
C	SET UP NEW INTERVAL.	00007134
	98 FILE(POINT,2)=FILE(CYCLE,2)	00007135
	FILE(POINT,3)=SUMQ	00007136
	IF (POINT.EQ.LEN) POINT=C	00007137
	POINT=POINT+1	00007138
	IF (I.EQ.LEN) GOTO 100	00007139
	CYCLE=K	00007140
	GOTO 96	00007141
	99 CYCLE=K	00007142
	SUMQ=SUMQ+FILE(K,3)	00007143
	GOTO 97	00007144
C	ALL INTERVALS ARE PROCESSED. ADJUST REMAINING PARAMETERS.	00007145
	100 FILEND=POINT-NFILE	00007146
	IF (FILEND.LT.1) FILEND=FILEND+LEN	00007147
	Z(1)=Z(12)	00007148
C	ISW1 = 2 TO SHOW THE FIRST TIME THROUGH WITH J CHANGED TO 2.	00007149
	ISW1=2	00007150
	GOTO 4	00007151
	END	00007152
C	00007153
	SUBROUTINE ACCELIZ,ERR,ERRR,J,I2,IFR	00007154
C	PURPOSE -	00007155
C	THIS ROUTINE USES THE SEQUENCE OF ESTIMATES OBTAINED AFTER EACH	00007156
C	CYCLE AND REPEATEDLY APPLIES THE ALTKEN DELTA-SQUARED TRANSFORM-	00007157
C	ATION TO OBTAIN NEW SEQUENCES. THE MOST ACCURATE VALUES OF THESE	00007158
C	ARE COMPARED TO DETERMINE IF CERTAIN CRITERIA HAVE BEEN MET.	00007159
C		00007160
C	DESCRIPTION OF PARAMETERS -	00007161
C	Z = ARRAY OF THE ORIGINAL SEQUENCE.	00007162
C	ERR - SEE SUBROUTINE FIRST.	00007163
C	ERRR - SEE SUBROUTINE AL9.	00007164
C	J - SEE SUBROUTINE INHAF.	00007165
		00007166

BEST AVAILABLE COPY

BEST AVAILABLE COPY

```

15 IF(KK.EQ.1) GOTO 50
   I=KK+1
   GOTO 10
C   MAKE SURE Z CONTAINS NO MORE THAN 19 ELEMENTS.
50 IF(IZ.NE.20) GOTO 65
   DO CO I=1,19
60 Z(I)=Z(I+1)
   IZ=IZ
65 RETURN
   END

```

```

00008270
00008280
00008290
00008300
00008310
00008320
00008330
00008340
00008350
00008360

```

BEST AVAILABLE COPY

ON THE OPTIMALITY OF THE RAYLEIGH-RITZ APPROXIMATION

S. C. Eisenstat, R. S. Schreiber, M. H. Schultz[†]

Department of Computer Science
Yale University
New Haven, Connecticut 06520

Abstract

Let A be a densely defined, self-adjoint positive definite linear operator on a Hilbert space H . Let u be the generalized solution of the equation $Au = f$, where $f \in H$, and let u_S be the Rayleigh-Ritz approximation to u from a finite-dimensional subspace S of the domain of A . We show that if the residual is bounded, i.e., there exists $B > 0$ such that, for every $g \in H$,

$$\|g - Av_S\| \leq B \|g\|$$

where v_S is the Rayleigh-Ritz approximation to the generalized solution of $Av = g$, then the Rayleigh-Ritz approximation is quasi-optimal in the sense that

$$\|u - u_S\| \leq B \inf_{v_S \in S} \|u - v_S\|.$$

For A a uniformly elliptic second-order differential operator and $\{S_h\}_{h>0}$ a suitable family of subspaces of the domain of A , we show that the residuals are bounded uniformly in h , so that the Rayleigh-Ritz approximations are uniformly quasi-optimal. In the case of a two-point boundary value problem, we remove the restriction that S_h be contained in the domain of A , and show that the Rayleigh-Ritz approximation is as good as any interpolate in S_h of the solution.

1. Introduction.

Let A be a densely defined, self-adjoint positive definite linear operator on a Hilbert space H , and let u be the generalized solution to the equation $Au = f$, where $f \in H$. Nitsche's trick (Nitsche [4]) has been widely used to prove optimal-order H -norm error bounds for the Rayleigh-Ritz method for approximating u (see Schultz [5] and the references therein). Our aim here is to establish a stronger result on the optimality of the Rayleigh-Ritz approximation.

[†]This research was supported in part by NSF Grant MCS 76-11450 and ONR Grant N00014-76-0277.

Let u_S be the Rayleigh-Ritz approximation to u in a finite-dimensional subspace S of the domain of A . In Section 2 we prove that the Rayleigh-Ritz approximation is quasi-optimal (in the H norm):

$$(1.2) \quad \|u - u_S\| < B \inf_{v_S \in S} \|u - v_S\|,$$

provided there exists a positive constant B such that, for every $g \in H$,

$$\|g - Av_S\| \leq B \|g\|,$$

where v_S is the Rayleigh-Ritz approximation to the generalized solution of $Av = g$. In Section 3 we show that if A is a uniformly elliptic second-order differential operator and $\{S_h\}_{h>0}$ is a suitable family of subspaces of the domain of A , then the residuals are bounded uniformly in h . Therefore, the Rayleigh-Ritz approximations are quasi-optimal uniformly in h . In particular, the result applies to subspaces of C^1 -piecewise polynomials. In Section 4 we consider self-adjoint two-point boundary value problems. Without requiring that S_h be a subspace of the domain of A , we show that the error in the Rayleigh-Ritz approximation is within a constant factor of the error in any interpolate in S_h of the solution.

2. H-norm error bounds.

Let H be a Hilbert space with inner product (\cdot, \cdot) and norm $\|\cdot\|$, and let A be a linear operator on H such that

- i) A is densely defined, i.e., the domain $D(A)$ of A is dense in H , and
- ii) A is self-adjoint and positive definite.

We define the inner product

$$[u, v] \equiv (Au, v), \quad u, v \in D(A),$$

and the corresponding norm

$$\|v\|_A \equiv [v, v]^{1/2} = (Av, v)^{1/2}, \quad v \in D(A).$$

The completion H_A of $D(A)$ with respect to $\|\cdot\|_A$ is itself a Hilbert space with inner product $[\cdot, \cdot]$. Furthermore,

$$(2.1) \quad [v, u] = (Av, u), \quad v \in D(A), \quad u \in H_A$$

(see Mikhlin [3]).

Let $f \in H$ and consider the linear equation

$$(2.2) \quad Au = f.$$

The generalized solution of (2.2) is the unique $u \in H_A$ satisfying

$$(2.3) \quad [u, v] = (f, v) \quad \text{for all } v \in H_A.$$

For every finite-dimensional subspace S of H_A , the Rayleigh-Ritz approximation u_S to u is the unique element of S satisfying

$$(2.4) \quad [u - u_S, v_S] = (f - Au_S, v_S) = 0 \quad \text{for all } v_S \in S.$$

We now give a general condition on the subspace S which is sufficient to make the Rayleigh-Ritz approximation quasi-optimal.

Theorem 1. Let S be a finite-dimensional subspace of $D(A)$ with the property that the residual in the Rayleigh-Ritz approximation is bounded; i.e., there exists a constant $B > 0$ such that, for every $g \in H$ the Rayleigh-Ritz approximation v_S to the generalized solution of $Av = g$ satisfies

$$(2.5) \quad \|g - Av_S\| \leq B \|g\|.$$

Then the Rayleigh-Ritz approximation is quasi-optimal in the H -norm in the sense that

$$(2.6) \quad \|u - u_S\| \leq B \inf_{v_S \in S} \|u - v_S\|.$$

Proof: Following Nitsche [4], let $e_S = u - u_S$, let $w \in H_A$ be the generalized solution of (2.2) with right-hand side e_S , and let $w_S \in S$ be the Rayleigh-Ritz approximation to w . By (2.4)

$$[e_S, w_S] = 0,$$

so that

$$(2.7) \quad \|e_S\|^2 = (e_S, e_S) = [w, e_S] = [w - w_S, e_S].$$

At this point in the argument, earlier authors applied the Cauchy-Schwarz inequality and used approximation-theoretic results to bound $\|w - w_S\|_A$ and $\|e_S\|_A$ (cf. [4], [5]). Instead, we continue, letting

\bar{u}_S denote an arbitrary element of S . By (2.4),

$$(2.8) \quad [w - w_S, u_S - \bar{u}_S] = 0.$$

Adding (2.8) to (2.7),

$$(2.9) \quad \begin{aligned} \|e_S\|^2 &= [w - w_S, u - \bar{u}_S] \\ &= [w, u - \bar{u}_S] - [w_S, u - \bar{u}_S]. \end{aligned}$$

By (2.3),

$$[w, v] = (e_S, v) \quad \text{for all } v \in H_A;$$

and since $S \subset D(A)$, we have by (2.1) that

$$[w_S, v] = (Aw_S, v) \quad \text{for all } v \in H_A.$$

Substituting these equalities in the right-hand side of (2.9),

$$\|e_S\|^2 = (e_S - Aw_S, u - \bar{u}_S).$$

Thus, by the Cauchy-Schwarz inequality and (2.5), we have that

$$\begin{aligned} \|e_S\|^2 &\leq \|e_S - Aw_S\| \|u - \bar{u}_S\| \\ &\leq B \|e_S\| \|u - \bar{u}_S\|. \end{aligned}$$

Since \bar{u}_S was arbitrary, this proves (2.6).

Q.E.D.

3. Strongly elliptic differential equations.

Let Ω be a bounded region in Euclidean n -space and consider the differential equation

$$(3.1) \quad Au \equiv - \sum_{i,j=1}^n \frac{\partial}{\partial x_i} (a_{ij} \frac{\partial u}{\partial x_j}) + bu = f,$$

subject to the Dirichlet boundary conditions

$$(3.2) \quad u = 0 \quad \text{on } \partial\Omega.$$

Let $H^S = H^S(\Omega)$ (respectively H_0^S) be the completion of the C^∞ functions (respectively the C^∞ functions with compact support in Ω) with respect to the norm

$$\|u\|_{H^s}^2 = \sum_{j=0}^s \int_{\Omega} (D^j u)^2 dx,$$

and let $\|\cdot\|$ denote the norm of $H^0 = L^2(\Omega)$. We assume that

$$i) a_{ij} = a_{ji}, \quad 1 \leq i, j \leq n,$$

$$ii) a_{ij} \in C^1(\Omega), \quad b \in C^0(\Omega),$$

iii) There exist positive constants λ and Λ such that

$$(3.3) \quad \lambda \|u\|_{H^1} \leq (Au, u)^{1/2} \leq \Lambda \|u\|_{H^1} \\ \text{for all } u \in H^2(\Omega) \cap H_0^1(\Omega).$$

With these hypotheses, $H_A = H_0^1(\Omega)$.

Let the generalized solution to (3.1)-(3.2) be defined as in Section 2. We assume the following regularity result:

iv) There exists a positive constant K such that, if $f \in L^2(\Omega)$, then the generalized solution u is in $H^2(\Omega)$ and

$$(3.4) \quad \|u\|_{H^2} \leq K \|f\|.$$

(For conditions under which this hypothesis is valid, see Schultz [5, p.103], Birman and Skvortsov [1], and Friedman [2, p.307].)

Let $\{S_h\}_{h>0}$ be a family of finite-dimensional subspaces of $D(A)$ which satisfy the following "approximation hypothesis": There exists a positive constant c_1 such that, for all $u \in H^2(\Omega)$ and all $h > 0$, there exists a $v_h \in S_h$ satisfying

$$(3.5) \quad \|u - v_h\|_{H^s} \leq c_1 h^{2-s} \|u\|_{H^2}, \quad s = 0, 1, 2.$$

We require further that S_h satisfy the "inverse assumption": There exists a positive constant c_2 such that, for all $h > 0$ and $v_h \in S_h$,

$$(3.6) \quad \|v_h\|_{H^2} \leq c_2 h^{-1} \|v_h\|_{H^1}.$$

These assumptions will be satisfied, for example, when S_h is a subspace of C^1 -piecewise polynomials of degree ≥ 2 with respect to a uniform mesh of size h .

We now show that the residual in the Rayleigh-Ritz approximation is uniformly bounded. We employ a technique of Stephens [6], who proved this result for the special case of Poisson's equation.

Theorem 2. If the operator A satisfies (i)-(iv) and the subspaces S_h satisfy the approximation hypothesis (3.5) and the inverse assumption (3.6), then the residuals $f - Au_h$ are bounded uniformly in h ; i.e., there exists a constant B independent of h such that, for every $f \in L^2$ and all $h > 0$, the Rayleigh-Ritz approximation $u_h \in S_h$ satisfies

$$(3.7) \quad \|f - Au_h\| \leq B \|f\|.$$

Proof: Because of the assumption (ii) and the smoothness of S_h , there exists a constant $M > 0$ such that

$$\|Au_h\| \leq M \|u_h\|_{H^2}.$$

Thus, in order to bound $\|f - Au_h\|$, it suffices to bound $\|u - u_h\|_{H^2}$ since

$$(3.8) \quad \begin{aligned} \|f - Au_h\| &\leq \|f\| + \|Au_h\| \\ &\leq \|f\| + M \|u_h\|_{H^2} \\ &\leq \|f\| + M \|u\|_{H^2} + M \|u - u_h\|_{H^2} \\ &\leq (1+MK) \|f\| + M \|u - u_h\|_{H^2} \end{aligned}$$

where we have used the regularity assumption to bound $\|u\|_{H^2}$.

Let v_h be an approximation to u which satisfies (3.5). By the triangle inequality,

$$(3.9) \quad \|u - u_h\|_{H^2} \leq \|u - v_h\|_{H^2} + \|v_h - u_h\|_{H^2}.$$

Because the Rayleigh-Ritz approximation is optimal in the norm of H_A , it follows from (3.3) and the approximation hypothesis that

$$(3.10) \quad \begin{aligned} \|u - u_h\|_{H^1} &\leq \lambda^{-1} \|u - u_h\|_A \\ &\leq \lambda^{-1} \|u - v_h\|_A \\ &\leq \lambda^{-1} \|u - v_h\|_{H^1} \\ &\leq c_1 \lambda^{-1} h \|u\|_{H^2}. \end{aligned}$$

Using the inverse assumption (3.6), (3.5) with $s = 1$, and (3.10),

$$\begin{aligned}
 (3.11) \quad \|v_h - u_h\|_{H^2} &\leq c_2 h^{-1} \|v_h - u_h\|_{H^1} \\
 &\leq c_2 h^{-1} (\|v_h - u\|_{H^1} + \|u - u_h\|_{H^1}) \\
 &\leq c_2 c_1 (1 + \lambda^{-1} \Lambda) \|u\|_{H^2}.
 \end{aligned}$$

Now, using (3.5) with $s = 2$ and (3.11) to bound the right-hand side of (3.9), and applying the regularity hypothesis (3.4), we have

$$\begin{aligned}
 (3.12) \quad \|u - u_h\|_{H^2} &\leq c_1 [1 + c_2 (1 + \lambda^{-1} \Lambda)] \|u\|_{H^2} \\
 &\leq K c_1 [1 + c_2 (1 + \lambda^{-1} \Lambda)] \|f\|.
 \end{aligned}$$

Finally, (3.7) follows from (3.8) and (3.12).

Q.E.D.

Corollary. Under the hypotheses of the theorem, there exists a constant B which depends only on the operator A such that, for all $h > 0$,

$$\|u - u_h\| \leq B \inf_{v_h \in S_h} \|u - v_h\|.$$

The result of the theorem also implies that the residual converges to 0.

Corollary. Under the hypotheses of the theorem, the functions Au_h converge to f in L^2 ; i.e.,

$$\|Au_h - f\| \rightarrow 0 \text{ as } h \rightarrow 0.$$

Proof: Stephens [6, Corollary 2.1] shows that Au_h converges to f if and only if $\|Au_h\| < B$ for some constant B independent of h .

Other conditions for the convergence to zero (and hence boundedness) of the residual have been obtained. See, for example, Mikhlin [3, p. 147].

4. The one-dimensional case.

In this section we consider the case of a second-order self-adjoint elliptic differential equation on the unit interval $I \equiv [0,1]$. We remove the requirement that the subspace S_h belong to the domain of the operator, and show that the Rayleigh-Ritz approximation is asymptotically as good as any interpolate of the solution in S_h .

Consider the two-point boundary value problem

$$(4.1) \quad \begin{aligned} Au &\equiv -D(a(x)Du) + b(x)u = f \quad \text{in } (0,1), \\ u(0) &= u(1) = 0 \end{aligned}$$

where $D \equiv \frac{d}{dx}$,

i) $a \in C^1(I)$ satisfies $a(x) \geq \delta > 0$ for all $x \in I$,

and

ii) $b \in C(I)$ satisfies $b(x) \geq 0$ for all $x \in I$.

Given $f \in L^2(I)$, the generalized solution $u \in H_0^1(I)$ is defined by

$$(4.2) \quad a(u,v) \equiv \int_0^1 (aDuDv + buv) dx = (f,v) \equiv \int_0^1 fv dx,$$

for all $v \in H_0^1(I)$. The form $a(u,v)$ is strongly coercive; i.e., there exists a constant K such that, for every $f \in L^2$ the generalized solution u is in H^2 and

$$\|u\|_{H^2} \leq K \|f\|$$

(see Schultz [5, p.103]).

Let S_h be a finite-dimensional subspace of H_0^1 with the property that there exists a partition $\Delta: 0 = x_0 < \dots < x_N = 1$ such that

$$(4.3) \quad \|Av_h\|_{\Delta}^2 \equiv \sum_{i=1}^N \int_{x_{i-1}}^{x_i} (Av_h)^2 dx < \infty$$

for all $v_h \in S_h$. This is essentially equivalent to requiring that S_h consist of continuous, piecewise- C^2 functions which satisfy the boundary conditions. We require that S_h satisfy a modified inverse assumption: There exists a positive constant c_2 such that, for all $h > 0$ and all $v_h \in S_h$,

$$(4.4) \quad \|v_h\|_{2,\Delta} \equiv \left(\sum_{i=1}^N \sum_{j=0}^2 \int_{x_{i-1}}^{x_i} (D^j v_h)^2 dx \right)^{1/2} \\ \leq c_2 h^{-1} \|v_h\|_{H^1}.$$

We also require that S_h satisfy a modified approximation hypothesis: There exists a positive constant c_1 such that, for all $u \in H^2$ and all $h > 0$, there exists a $v_h \in S_h$ satisfying

$$(4.5) \quad \|u - v_h\|_{H^s} \leq c_1 h^{2-s} \|u\|_{H^2}, \quad s = 0, 1$$

and

$$(4.6) \quad \|u - v_h\|_{2,\Delta} \leq c_1 \|u\|_{H^2}.$$

Finally we require that S_h satisfy the following interpolation hypothesis: Given $\underline{d} \in \mathbb{R}^{N-1}$ there exists an element $v_h \in S_h$ satisfying

$$(4.7) \quad v_h(x_i) = d_i, \quad 1 \leq i \leq N-1.$$

For example, these hypotheses are satisfied when S_h is a subspace of continuous piecewise polynomials of degree ≥ 1 with respect to a uniform mesh Δ of size h .

Theorem 3. Let u be the generalized solution of (4.1) with $f \in L^2$. Let S_h be a subspace of H_0^1 satisfying (4.3), the inverse assumption (4.4), the approximation hypothesis (4.5)-(4.6), and the interpolation hypothesis (4.7). Let $u_h \in S_h$ be the Rayleigh-Ritz approximation to u . There exists a constant B independent of h , f , and u such that

$$\|u - u_h\| \leq B \inf_{\substack{v_h \in S_h \\ v_h \text{ interpolates } u}} \|u - v_h\|.$$

Proof: As in the proof of Theorem 1, let $e_h = u - u_h$, let w be the generalized solution to (4.1) with right-hand side e_h , and let w_h be the Rayleigh-Ritz approximation to w in S_h . Let $v_h \in S_h$ interpolate u at the knots of Δ ; i.e., v_h satisfies (4.7) with $d_i = u(x_i)$. By the same argument that leads to (2.9), we obtain

$$(4.8) \quad \|e_h\|^2 = a(w, u - v_h) - a(w_h, u - v_h).$$

Integrating by parts,

$$\begin{aligned} a(w_h, u - v_h) &= \sum_{i=1}^N \int_{x_{i-1}}^{x_i} (aDw_h D(u - v_h) + bw_h(u - v_h)) dx \\ &= \sum_{i=1}^N \left(aDw_h(u - v_h) \Big|_{x_{i-1}}^{x_i} + \int_{x_{i-1}}^{x_i} [-D(aDw_h) + bw_h](u - v_h) dx \right). \end{aligned}$$

But by the interpolation conditions (4.7), the integrated terms vanish. Thus by (4.8) and (4.2)

$$\begin{aligned} \|e_h\|^2 &= a(w, u - v_h) - \sum_{i=1}^N \int_{x_{i-1}}^{x_i} [-D(aDw_h) + bw_h](u - v_h) dx \\ &= (e_h, u - v_h) - \sum_{i=1}^N \int_{x_{i-1}}^{x_i} Aw_h(u - v_h) dx \end{aligned}$$

(4.9)

$$= \sum_{i=1}^N \int_{x_{i-1}}^{x_i} (e_h - Aw_h)(u - v_h) dx$$

$$\leq \|e_h - Aw_h\|_{\Delta} \|u - v_h\|.$$

We now bound the norm of the residual $\|e_h - Aw_h\|_{\Delta}$ by the norm of the right-hand side $\|e_h\|$. The proof is identical to that of Theorem 2, except that we use the $\|\cdot\|_{2,\Delta}$ norm instead of the $\|\cdot\|_{H^2}$ norm; we omit the details. Together with (4.9) this yields

$$\|e_h\| \leq B \|u - v_h\|.$$

Since all we assumed about v_h was that it interpolates u , the proof is finished.

Q.E.D.

References

1. M. S. Birman and G. E. Skvortsov. On the summability of the highest-order derivatives of the solution of the Dirichlet problem in a domain with piecewise smooth boundary. Izv. Vyssh. Nchebn. Zaved Matimatika, 30(1962), pp. 12-21.
2. A. Friedman. Partial Differential Equations of Elliptic Type. Prentice Hall, Englewood Cliffs, N.J., 1964.
3. S. G. Mikhlin. The Problem of the Minimum of a Quadratic Functional. Holden-Day, San-Francisco, 1965.
4. J. Nitsche. Ein kriterium fur die quasi-optimalitat des Ritzschen verfahrens. Numer. Math., 11(1968), pp. 346-348.
5. M. H. Schultz. Spline Analysis. Prentice-Hall, Englewood Cliffs, N.J., 1973.
6. A. B. Stephens. The convergence of the residual for projective approximations. SIAM J. Numer. Anal., 10(1976), pp. 607-614.

Added in proof: We wish to thank Ivo Babuska for pointing out that a result similar to Theorem 1 appears in

Ivo Babuska and A. K. Aziz. Survey lectures on the mathematical foundations of the finite element method. In A. K. Aziz, editor, The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, 3-359. Academic Press, 1972.

NUMERICAL SOLUTION OF GUN TUBE PROBLEMS
IN THE ELASTIC-PLASTIC RANGE

Peter C. T. Chen
Benet Weapons Laboratory
Watervliet Arsenal
Watervliet, New York 12189

ABSTRACT. A finite element computer program for treating axisymmetric, elastic-plastic boundary value problems is described. Quadrilateral ring elements are used. The material is assumed to obey the Mises yield criterion and the Prandtl-Reuss flow rule. In order to test the accuracy of the program the elastic-plastic deformation in a pressurized gun tube is investigated and the results are compared with a finite-difference solution. The applicability of the program to two-dimensional axisymmetric problems is demonstrated for a gun tube of finite length loaded over part of its inner surface.

1. INTRODUCTION. Of all the available elastic-plastic solutions, the problem of pressurized thick-walled tubes has received the greatest attention. This is because of the symmetric nature of the problem and its practical importance to pressure vessels and the autofrettage process of gun barrels. Many investigations for the one dimensional problem have been reported over the last two decades. However, some problems are still too difficult to be solved analytically (ref. 1). If we try to solve a two dimensional elastic-plastic problem involving partial differential equations, the chance of success is even more remote. With the recent development in the finite element technique and high speed computer, numerical solutions to two- and three-dimensional elastic-plastic problems can now be obtained (ref. 2,3,4). Many good computer programs such as the MARC system have been developed, but they are not available for general distribution.

In this paper, the incremental tangent-modulus approach of the finite element formulation together with the computer program will be described. The material is assumed to obey the Mises yield criterion and the Prandtl-Reuss flow rule. Quadrilateral ring elements will be used to solve one dimensional as well as two dimensional gun tube problems in the elastic-plastic range. The one dimensional elastic-perfectly-plastic tube problem was solved for the purpose of evaluating the convergence and accuracy of the present program. The two dimensional elastic-plastic strain-hardening tube was solved for demonstrating the applicability of the program. Typical results are shown graphically. This approach is quite general. The elastic-plastic material can be strain hardening or non-hardening. The limitation to the use of strain hardening material such as in NASTRAN (ref. 5) is not necessary.

2. FINITE ELEMENT FORMULATION. The variational principle for elastic-plastic solid (ref. 6) can be used as a theoretical basis for the finite element formulation. In the absence of body forces and considering only quasi-static deformation, the principle states that among all admissible displacement-increment vectors, $\{\Delta u\}$, the actual one renders the following functional stationary

$$\phi = \frac{1}{2} \int_V \{\Delta \sigma\}^T \{\Delta \epsilon\} dv - \int_{s_f} \{\Delta u\}^T \{\Delta f\} ds \quad (1)$$

where $\{\Delta \sigma\}$ is the stress-increment vector; $\{\Delta \epsilon\}$, the strain-increment vector; $\{\Delta u\}$, the displacement-increment vector; and $\{\Delta f\}$, the traction-increment vector prescribed over a portion of the boundary s_f . The superscript T denotes the transpose. The stress-increment vector is related to the strain-increment vector which is derivable from an admissible displacement-increment vector.

An elastic-plastic body is divided into L elements interconnected at a finite number of N points. If $\phi^{(\ell)}$ is the functional (1) with respect to the ℓ th element, then

$$\phi = \sum_{\ell=1}^L \phi^{(\ell)} \quad (2)$$

The elemental functional $\phi^{(\ell)}$ can be evaluated approximately in the following manner. First, an approximation is made concerning the incremental displacement, $\{\Delta u\}$, within an individual finite element in terms of discrete quantities at nodal points, $\{\Delta U\}$. This relation can be expressed in the form

$$\{\Delta u\} = [N] \{\Delta U\} \quad (3)$$

in which the components of $[N]$ are in general functions of position and $\{\Delta U\}$ satisfy the compatibility conditions within and along the boundary of adjacent elements.

The strain-increment vector, $\{\Delta \epsilon\}$, is derivable from the assumed displacement-increment vector, $\{\Delta u\}$, and can be expressed in the form

$$\{\Delta \epsilon\} = [B] \{\Delta U\} \quad (4)$$

where $[B]$ is a kinematical matrix.

The incremental stress-strain relations for an elastic-plastic solid which obeys Mises' yield condition, Prandtl-Reuss flow rule and isotropic hardening rule can be expressed in closed form (ref. 2).

$$\{\Delta \sigma\} = [D] \{\Delta \epsilon\} \quad (5)$$

In the two-dimensional axisymmetric case, the constitutive matrix [D] can be written as

$$[D] = [DE] - [DP] \quad (6)$$

with

$$[DE] = \frac{2G}{1-2\nu} \begin{bmatrix} 1-\nu & & & \text{SYM.} \\ \nu & 1-\nu & & \\ \nu & \nu & 1-\nu & \\ 0 & 0 & 0 & \frac{1}{2}(1-2\nu) \end{bmatrix} \quad (7)$$

and

$$[DP] = \frac{2G}{s} \begin{bmatrix} \sigma_Y'^2 & & & \text{SYM.} \\ \sigma_Y'\sigma_\theta' & \sigma_\theta'^2 & & \\ \sigma_Y'\sigma_z' & \sigma_\theta'\sigma_z' & \sigma_z'^2 & \\ \sigma_Y'T_{Yz} & \sigma_\theta'T_{Yz} & \sigma_z'T_{Yz} & T_{Yz}^2 \end{bmatrix} \quad (8)$$

where

$$\sigma_m = \frac{1}{3} (\sigma_Y + \sigma_\theta + \sigma_z), \quad \sigma_Y' = \sigma_Y - \sigma_m, \text{ etc.}$$

$$s = \frac{2}{3} \bar{\sigma}^2 \left[1 + \frac{2}{3} (1+\nu)\omega/(1-\omega) \right]$$

$$\bar{\sigma}^2 = \frac{1}{2} [(\sigma_Y - \sigma_\theta)^2 + (\sigma_\theta - \sigma_z)^2 + (\sigma_z - \sigma_Y)^2 + 6T_{Yz}^2]$$

$$\omega = E_t/E, \quad G = \frac{1}{2} E/(1+\nu)$$

and E_t is the tangent-modulus, i.e. the slope of the effective stress-strain curve obtained from a tension test or from a torsion test, as shown approximately in Fig. 2.

The elemental function $\phi^{(l)}$, upon substitution of equations (3), (4) and (5) into the functional (1), becomes

$$\phi^{(l)} = \frac{1}{2} \{\Delta U\}^T [K]^{(l)} \{\Delta U\} - \{\Delta U\}^T \{\Delta F\}^{(l)} \quad (9)$$

where

$$[K]^{(l)} = \int_V [B]^T [D] [D] dv \quad (10)$$

is the element stiffness matrix and

$$\{\Delta F\}^{(l)} = \int_{S_f} [N]^T \{\Delta f\} ds_f \quad (11)$$

is the nodal point force-increment vector over the subregion. The global stiffness matrix $[K]$, and the nodal force-increment vector, $\{\Delta Q\}$, are the sums of those of the subregions. The necessary condition for the functional to assume a stationary value gives the following stiffness equation

$$[K] \{\Delta q\} = \{\Delta Q\} \quad (12)$$

where $\{\Delta q\}$ is the global displacement-increment vector at all nodal points.

3. QUADRILATERAL RING ELEMENT. Consider a quadrilateral ring element which consists of four triangular ring elements, as shown in Figure 1. The coordinates of a fictitious nodal point 5 are chosen to be the average of those of nodal points 1, 2, 3 and 4. By assuming a linear distribution of the displacement-increments within each triangular ring element and determining the coefficients such that the distribution passes through specified nodal points, the stiffness matrix for each triangular ring element is first obtained (ref. 4). To formulate the stiffness matrix of the quadrilateral ring element, the nodal force-increment vectors for four triangular ring elements are assembled, and the resulting equations are partitioned in the form

$$\begin{bmatrix} [K_{aa}] & [K_{ab}] \\ [K_{ba}] & [K_{bb}] \end{bmatrix} \begin{Bmatrix} \{\Delta U_a\} \\ \{\Delta U_b\} \end{Bmatrix} = \begin{Bmatrix} \{\Delta F_a\} \\ \{\Delta F_b\} \end{Bmatrix} \quad (13)$$

where

$$\begin{aligned} \{\Delta U_a\}^T &= [\Delta U_{11}, \Delta U_{12}, \Delta U_{21}, \Delta U_{22}, \Delta U_{31}, \Delta U_{32}, \Delta U_{41}, \Delta U_{42}] \\ \{\Delta F_a\}^T &= [\Delta F_{11}, \Delta F_{12}, \Delta F_{21}, \Delta F_{22}, \Delta F_{31}, \Delta F_{32}, \Delta F_{41}, \Delta F_{42}] \\ \{\Delta U_b\}^T &= [\Delta U_{51}, \Delta U_{52}], \text{ and } \{\Delta F_b\}^T = [\Delta F_{51}, \Delta F_{52}] \end{aligned} \quad (14)$$

Eliminating the nodal point variables $\{\Delta U_b\}$ from equation (13), the stiffness matrix $[K]^L$ of the quadrilateral ring element becomes

$$[K]^L = [K_{aa}] - [K_{ab}][K_{bb}]^{-1}[K_{ba}] \quad (15)$$

where $[K_{bb}]^{-1}$ denotes the inverse of the matrix $[K_{bb}]$. After determining the nodal displacement-increment vectors of the element, the strain-increment vectors in each triangular ring can be calculated, and the average of the strain-increment values of four triangular rings evaluated at nodal point 5 is assigned as the value of the strain-increment vector of the quadrilateral ring element.

4. COMPUTER PROGRAM AND EVALUATION. A digital computer program for solving axisymmetric elastic-plastic boundary value problems was developed on the basis of triangular ring elements (ref. 4). The existing finite element program has been modified so that both triangular and quadrilateral ring elements can be used in modelling the complete structure. The sequence of the present program is similar to that of (ref. 2) for plane stress problems. The computer used is IBM 360 Model 44. The overlay feature has been utilized for reducing the core storage requirement. The load-increments can be prescribed or determined by scaling to cause at least one more element to become yielded. Another feature of this program is its capability of restarting. This enables the user to restart a program from a point of completion of a given loading sequence. This is because all previous results were written on tape. In addition to its restart capability, the results on tape can be used for output plotting.

In order to evaluate the convergence and accuracy of the computer program, the plane-strain problem of elastic-plastic thick-walled tube under uniform internal pressure was first investigated. The tube of outside radius 2" and inside radius 1" has been divided into 2, 5, 10, 20 and 25 quadrilateral ring elements, respectively; and the five-element model is shown in Figure 3. The numerical results showing the relation between internal pressure and inside radial displacement are given in Figure 4. The effect of reducing the element size on the rate of convergence is quite significant. The differences between 20 and 25 elements are negligible and too small to be shown in the figure. Both applied and scaled loading approaches have been used. The number of loading steps for the scaled loading approach is equal to the number of elements. To achieve the same accuracy by using both approaches, we need a lot of more steps for the applied loading approach. The results shown in Figure 4 are based on the scaled loading approach. Another way of evaluating this program is by comparing the present solution with an exact solution due to Hodge and White (ref. 7). That solution was based on the finite-difference method and the stress results were given for an elastic-perfectly-plastic tube with $b/a = 2$ and $\rho/a = 1.5$. The results by the finite element method for the radial, tangential and axial stress distribution are shown in Figures 5, 6 and 7, respectively, with $b/a = 2.0$ and $\rho/a = 1.02, 1.26, 1.50, 1.74, 1.98$. By comparing the present results with those in (ref. 7) we can conclude that the 25-element model does converge correctly. This is another indication of the accuracy of the present approach.

5. A TWO-DIMENSIONAL TUBE. The formulation and computer program developed here can be used for analyzing axisymmetric body of revolution under axisymmetric mechanical loads. The illustrated example is a two-dimensional elastic-plastic thick-walled tube problem as shown in Figure 8. The tube with inner radius (1"), outer radius (2") and length (4") is loaded uniformly over a middle portion (2") of the inner surface. The mesh generation and the loading for the half of the undeformed structure is shown in the figure. The tube material is assumed to be elastic-plastic strain hardening with the following properties: $E = 3 \times 10^7$ psi, $\nu = 0.3$, $\sigma_y = Y_1 = 1.5 \times 10^5$ psi, $\omega_1 = 0.05$, $Y_2 = 2.25 \times 10^5$ psi, $\omega_2 = 0.0$. This problem was solved in (ref. 8) for a non-hardening material based on a different formulation and different loading approach. For the present problem, we have determined the elastic solution at the moment that the first element becomes plastic; the corresponding pressure is found to be $.5676 \sigma_y$. We then use the scaled incremental loading approach until one of the outside element becomes yielded. Ten additional cycles were needed and the sequence in which the elements becomes plastic is 1,5,9,2,13,6,10,3,7,14,11,17,4. Some cycle will cause more than one element to become plastic. This is because those elements with effective stress $\bar{\sigma} \geq 0.99 Y_1$ have been considered as plastic. This allowance is advantageous for saving computing time. It takes one minute CPU time per cycle for the present problem on our computer (IBM 360 Model 44). The numerical results for the radial displacement at the inside, U_a (point 1) and outside, U_b (point 5) as functions of internal pressure are shown in Figure 9. The radial displacements along the bore surface for various stages of loading are shown in Figure 10. Finally the stresses at the centroid of one inside element (No. 1) are shown in Figure 11. The effect of loading history on the four stress components can be seen from the figure.

REFERENCES

1. Davidson, T. E. and Kendall, D. P., "The Design of Pressure Vessels for Very High Pressure Operations," Watervliet Arsenal Report WVT-6917, Watervliet, New York (1969). Also in Mechanical Behavior of Materials Under Pressure (edited by Pugh, H. L. D.), Elsevier Company (1970), Chapter 2.
2. Yamada, Y., Yoshimura, N., and Sakurai, T., "Plastic Stress-Strain Matrix and Its Application for the Solution of Elastic-Plastic Problems by the Finite Element Method," Int. J. Mech. Sci. Vol. 10 (1968), pp. 343-354.
3. Chen, P. C. T., "The Finite Element Analysis of Elastic-Plastic Thick-Walled Tubes," Proceedings of the Army Symposium on Solid Mechanics, 1972 - The Role of Mechanics in Design - Ballistic Problems, AMMRC MS 73-2, pp. 243-253 (1973).
4. Chen, P. C. T., "Elastic-Plastic Solution of a Two-Dimensional Tube Problem by the Finite Element Method," Transactions of the Nineteenth Conference of Army Mathematicians, ARO Report 73-3, pp. 763-784 (1973).
5. The NASTRAN Theoretical Manual NASA SP-221(01), (1972), NASA, Washington, D.C.
6. Hodge, P. G., "The Mathematical Theory of Plasticity," in Elasticity and Plasticity, John Wiley & Sons, Inc. (1958).
7. Hodge, P. G. and White, G. N., "A Quantitative Comparison of Flow and Deformation Theories of Plasticity," J. Appl. Mech. Vol. 72 (1950), pp. 180-184.
8. Meijers, P., "Elastic-Plastic Deformation of Thick-Walled Cylinders," First Int'l Conf. on Pressure Vessel Technology, Part 1, Design and Analysis, ASME (1969), pp. 13-34.

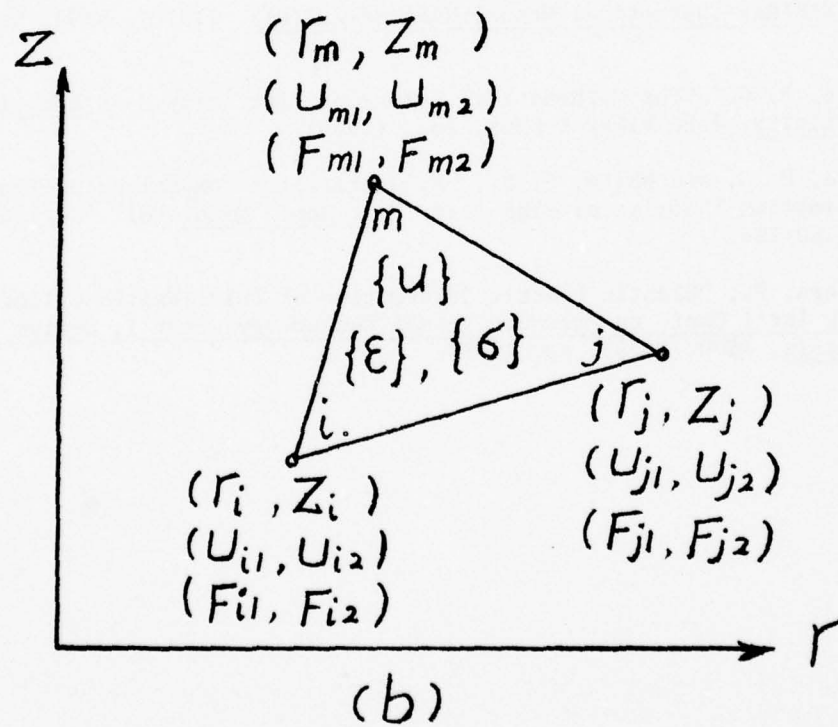
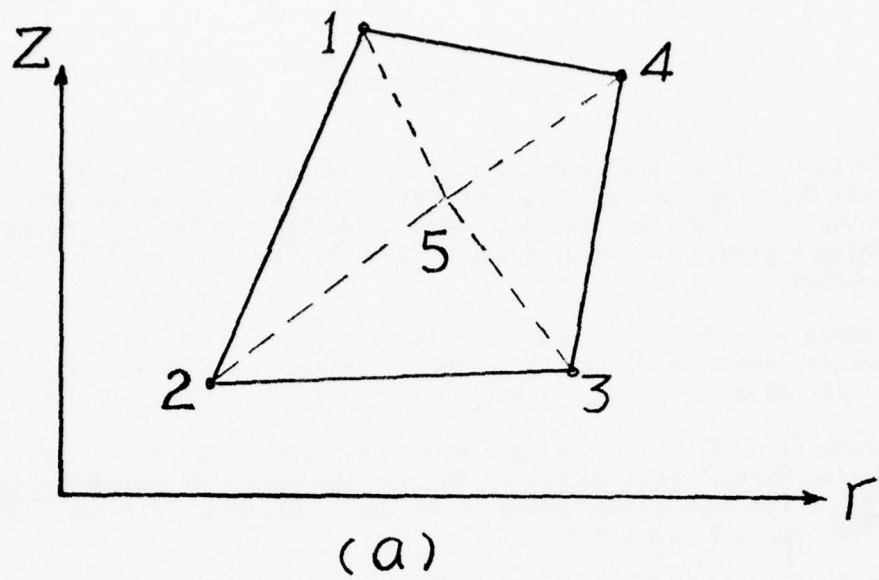


FIGURE 1. QUADRILATERAL RING ELEMENT

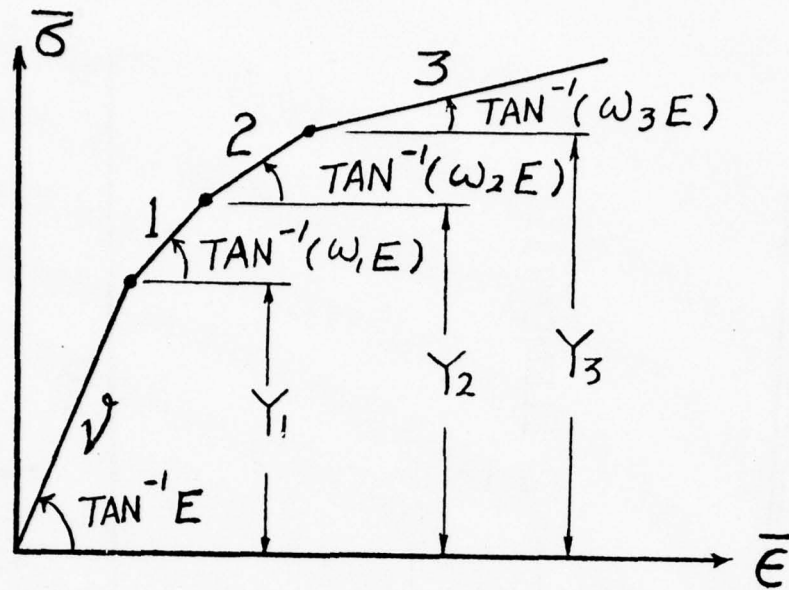


FIGURE 2. EFFECTIVE STRESS-STRAIN CURVE

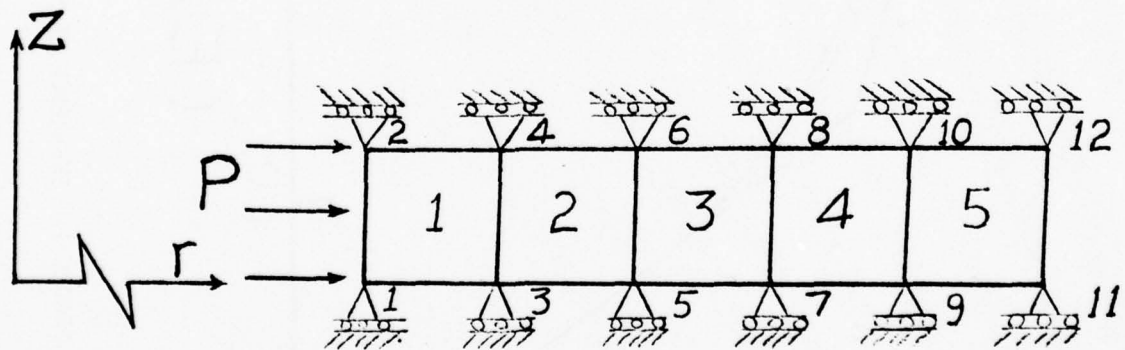


FIGURE 3. FIVE-ELEMENT THICK-WALLED TUBE UNDER UNIFORM PRESSURE

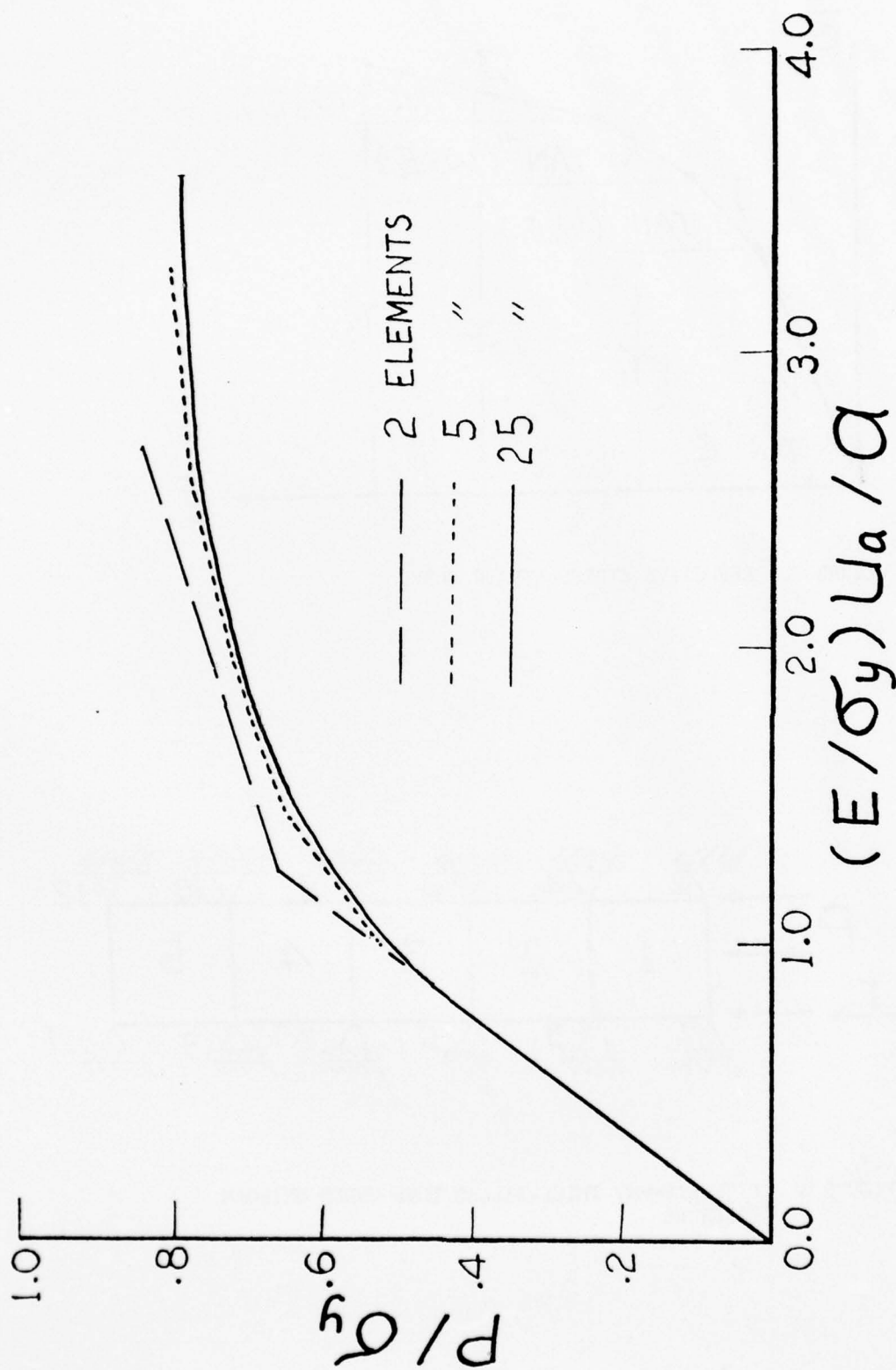


FIGURE 4. INFLUENCE OF NUMBER OF ELEMENTS AND LOADING STEPS

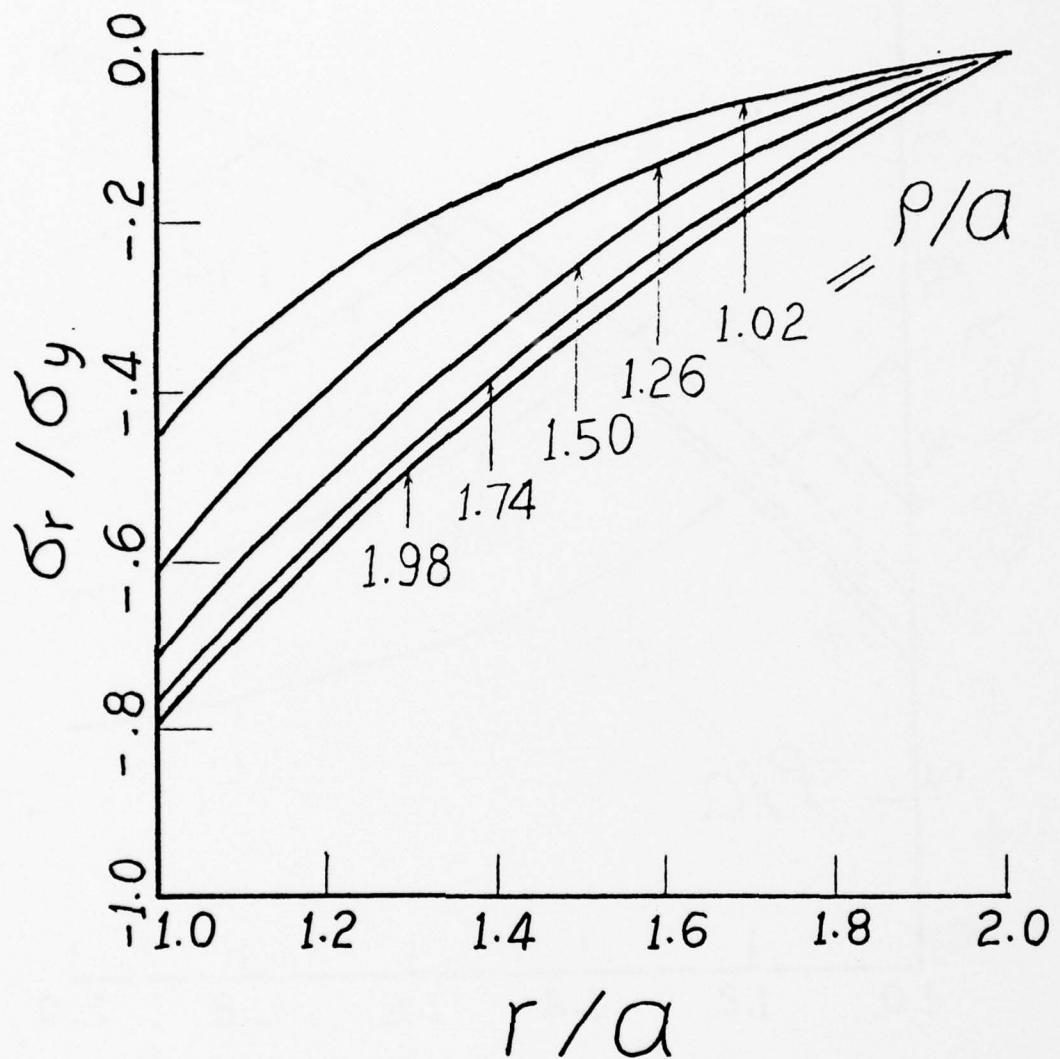


FIGURE 5. RADIAL STRESS IN AN ELASTIC-PERFECTLY-PLASTIC TUBE

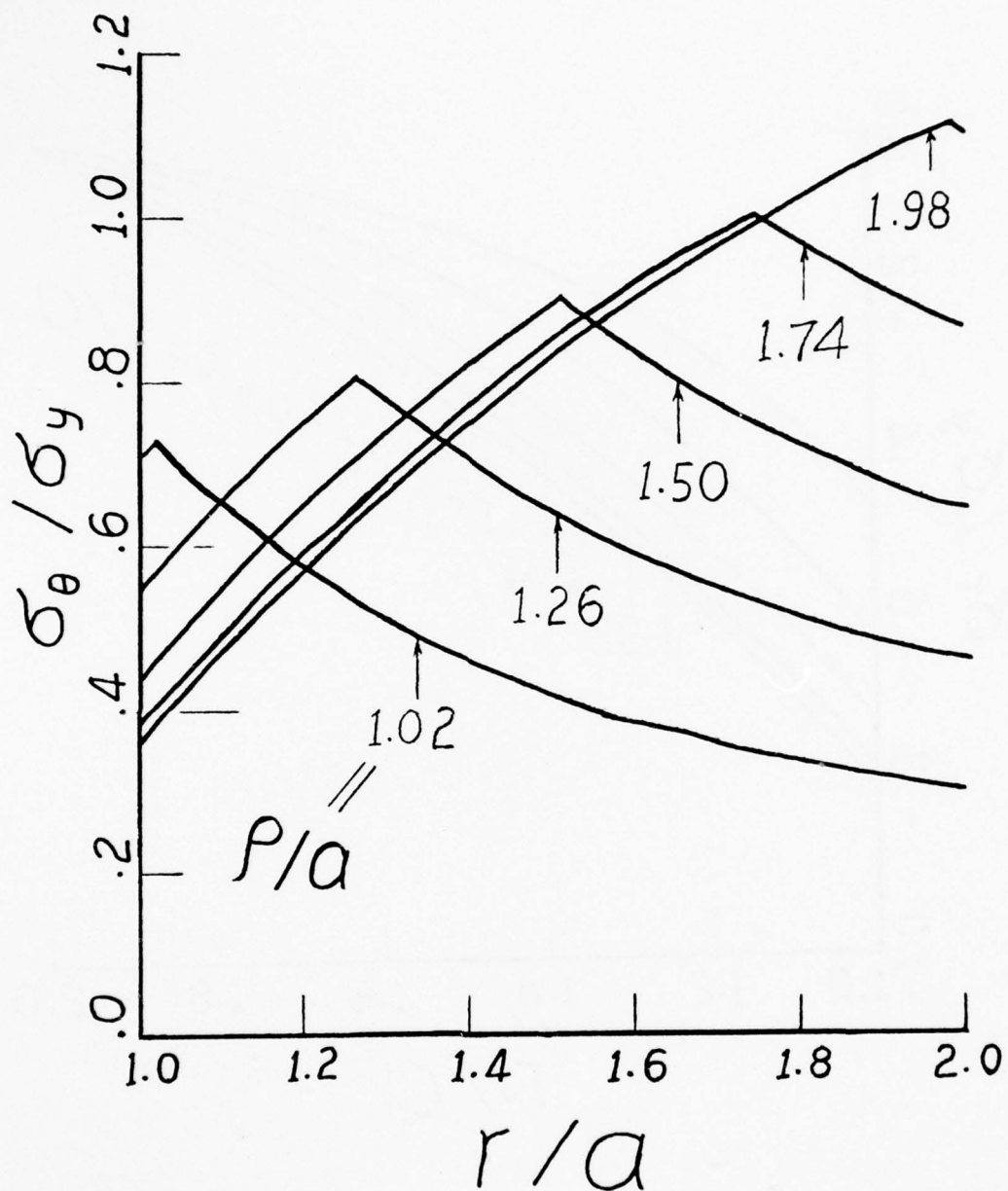


FIGURE 6. TANGENTIAL STRESS IN AN ELASTIC-PERFECTLY-PLASTIC TUBE

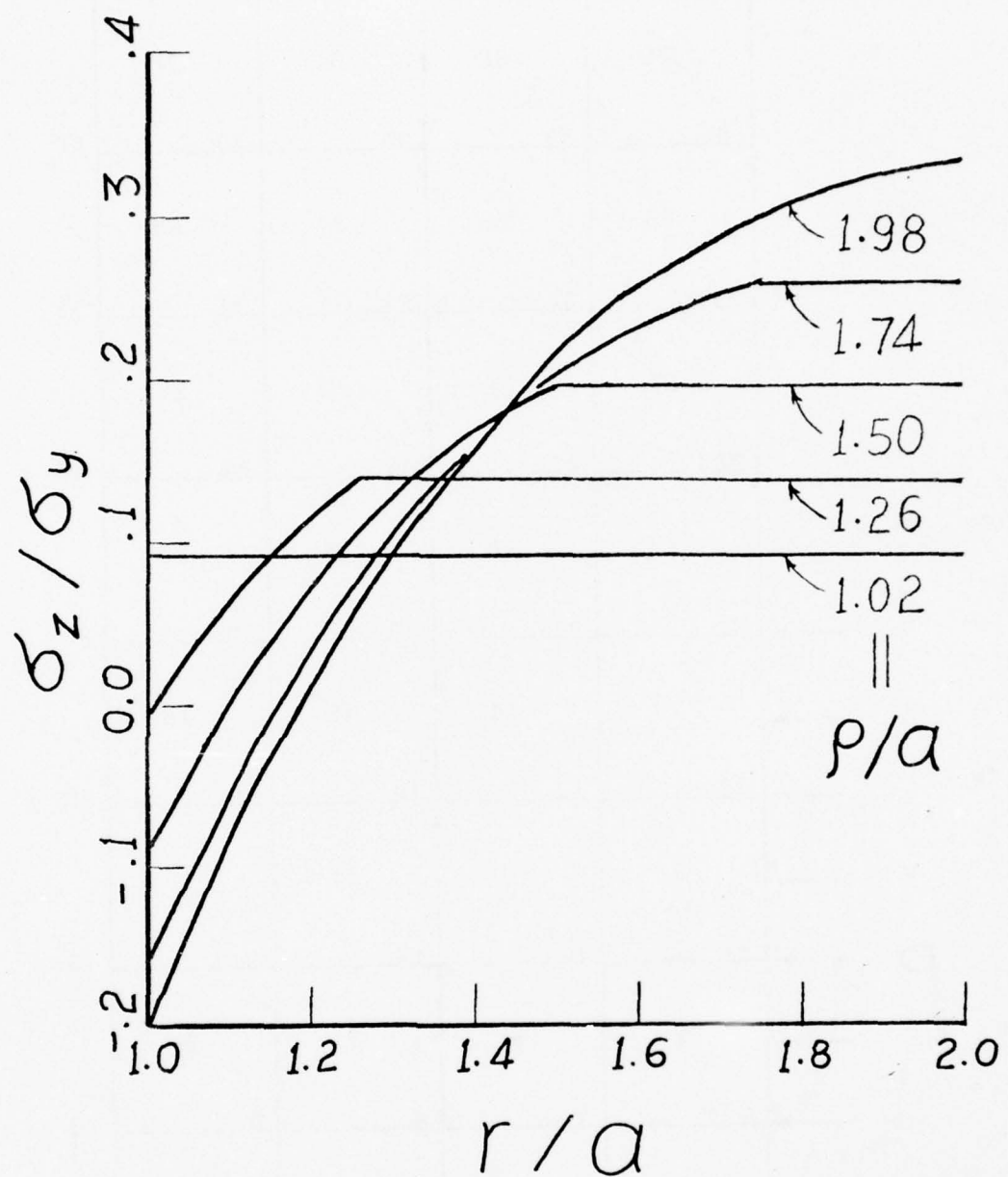


FIGURE 7. AXIAL STRESS IN AN ELASTIC-PERFECTLY-PLASTIC TUBE

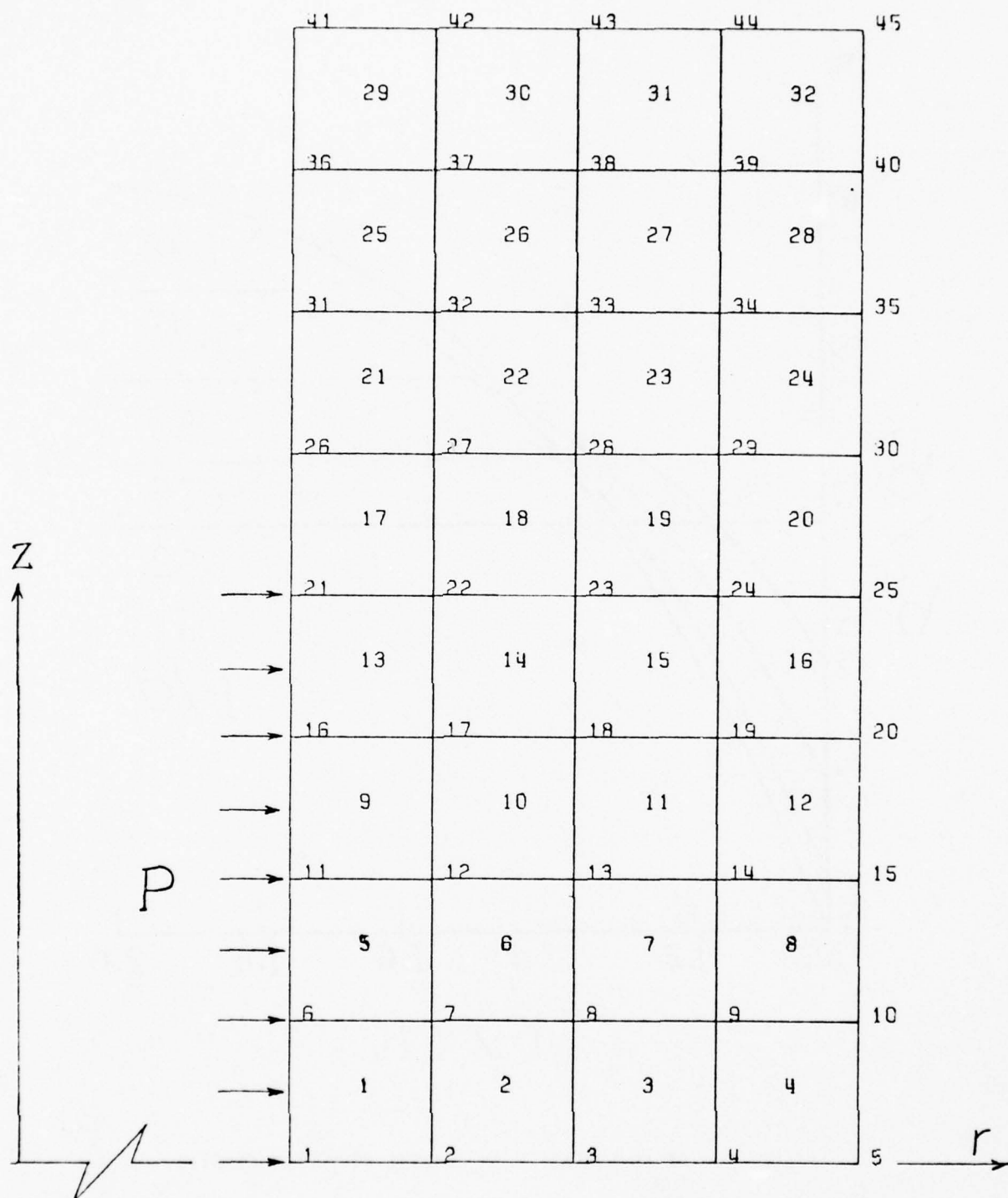


FIGURE 8. TWO DIMENSIONAL ELASTIC-PLASTIC TUBE

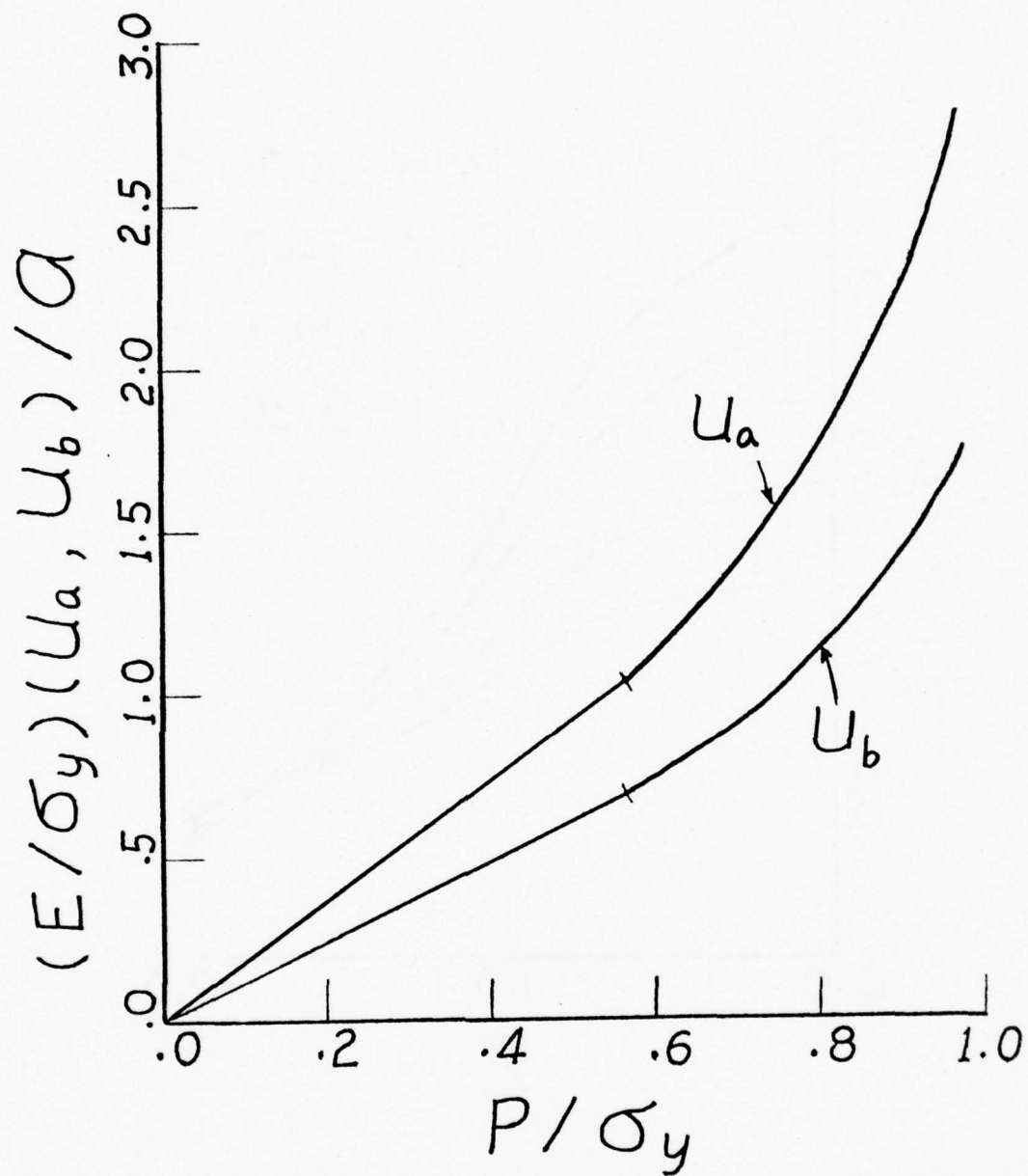


FIGURE 9. RADIAL DISPLACEMENTS AS FUNCTIONS OF INTERNAL PRESSURE

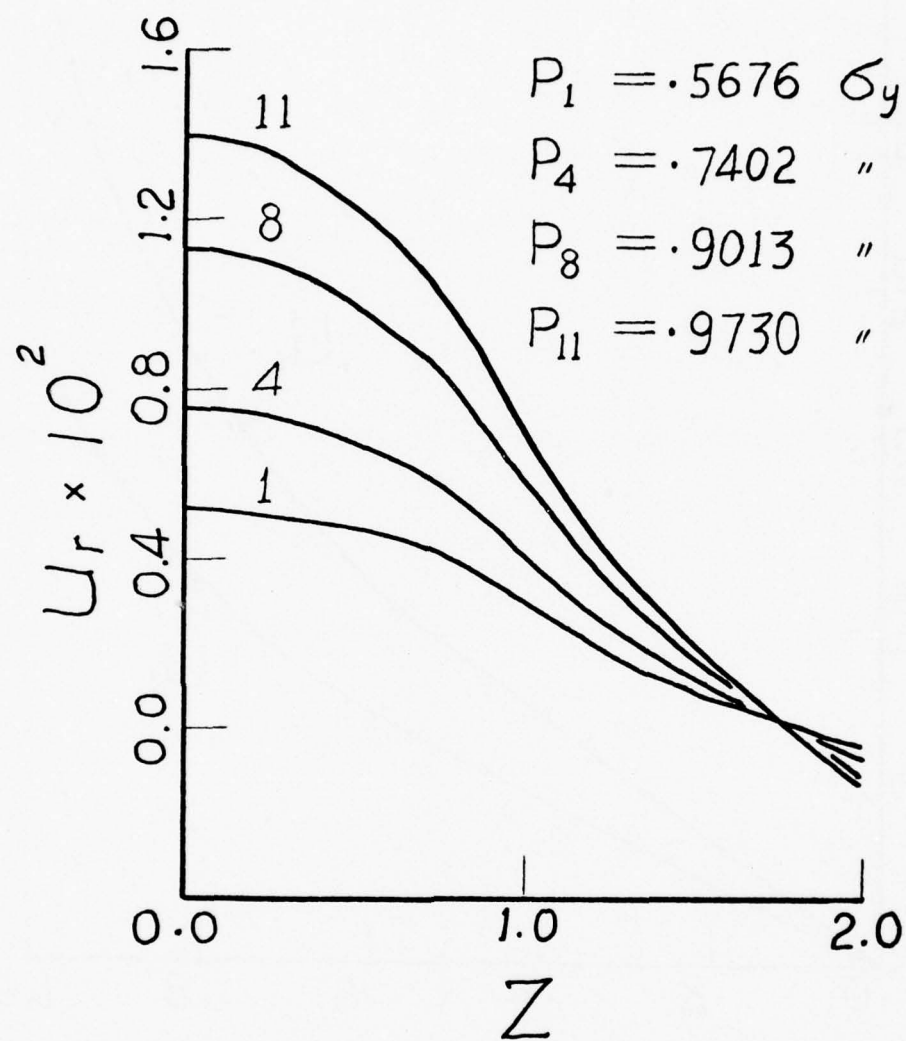


FIGURE 10. RADIAL DISPLACEMENT ALONG THE BORE

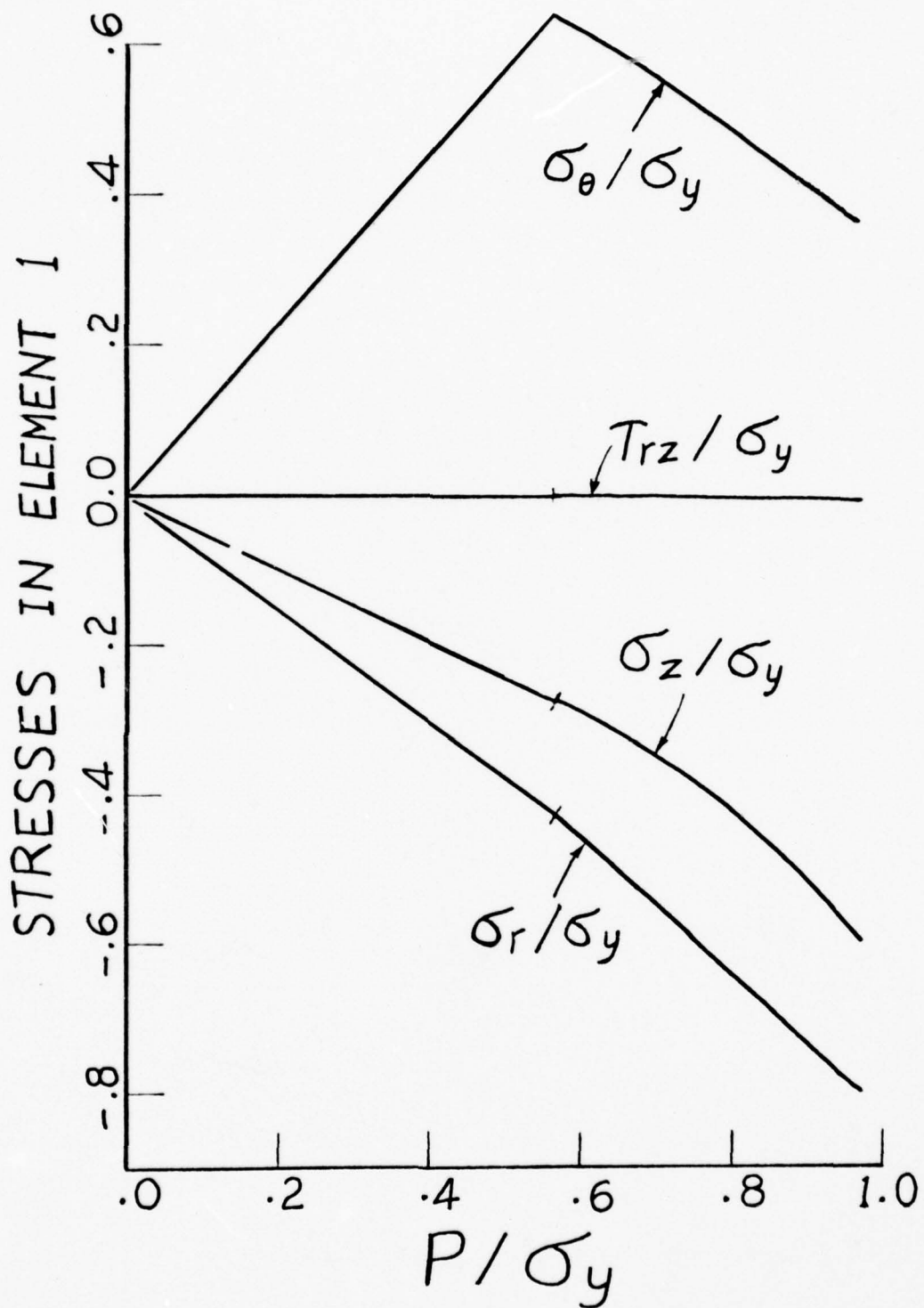
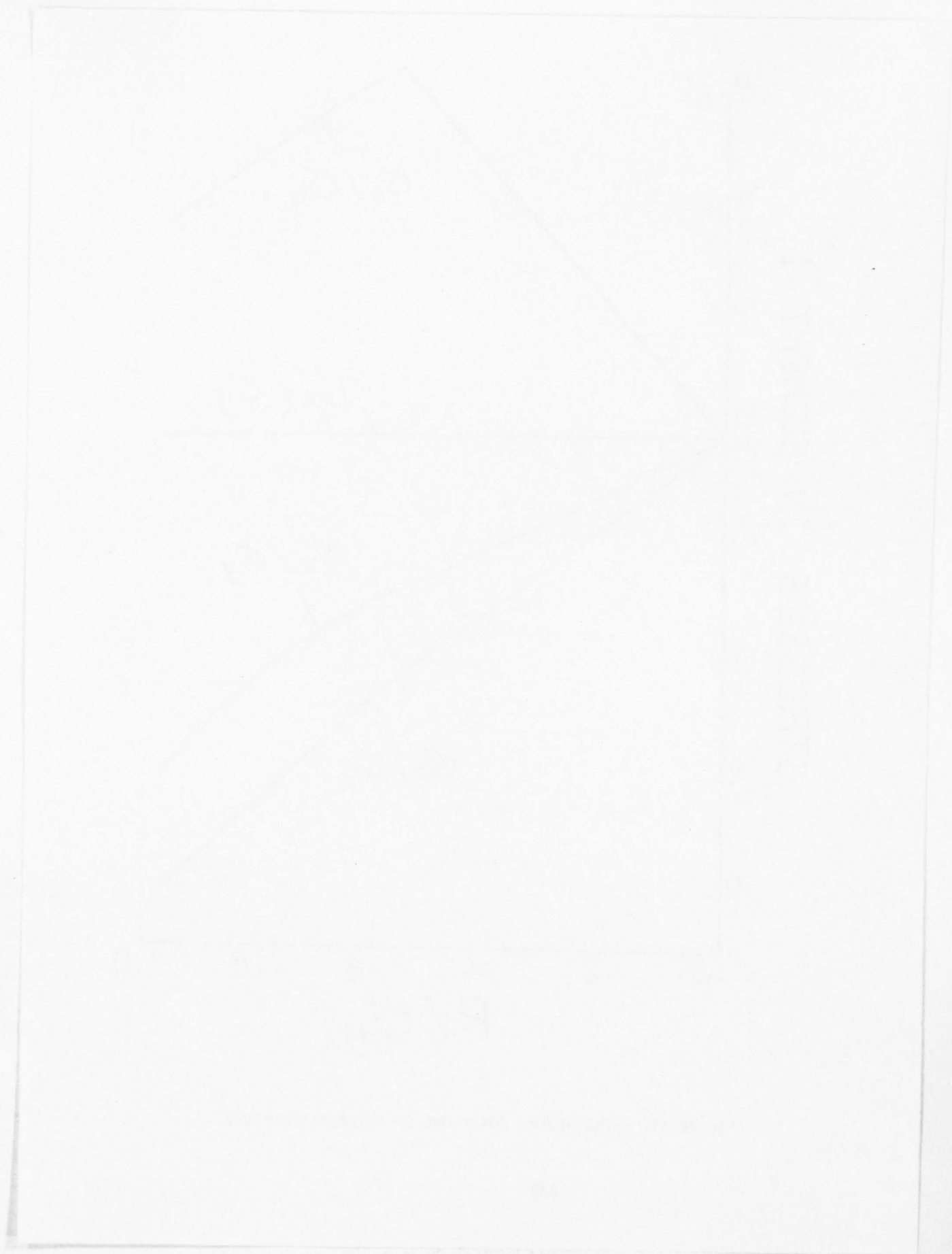


FIGURE 11. STRESSES AS FUNCTIONS OF INTERNAL PRESSURE



PERTURBATION METHODS FOR THE SOLUTION OF LINEAR PROBLEMS

L. B. Rall

Mathematics Research Center
University of Wisconsin-Madison

Dedicated to Professor Arvid T. Lonseth on his 65th Birthday

Abstract. Linear problems of central interest in numerical analysis are the solution of linear equations, the construction of the inverse or a generalized inverse of a linear operator, finding the eigenvalues and eigenvectors of a linear operator, and linear programming. A survey is made of methods which apply if the data of a solved linear problem is perturbed by operators and vectors of small norm (analytic perturbation), or by operators of finite rank and vectors belonging to a finite-dimensional subspace (algebraic perturbation). Perturbation methods may be used to extend the theory of linear problems, to estimate errors due to inaccurate data and computation, and to solve perturbed problems with economy of effort.

1. Linear problems. In the abstract framework of functional analysis, a linear problem is one which can be formulated in terms of linear spaces and operators [38, Chapter I]. Naturally, many problems of theoretical and practical interest in numerical analysis belong to this general class. Among these problems, some are important enough to be the subjects of extensive investigations, and also appear in the daily workload of most computing centers devoted to general scientific computation. Of these significant problems, the ones singled out for discussion here are: (a) solution of linear equations, (b) inversion of linear operators, (c) finding the eigenvalues and eigenvectors of a linear operator, and (d) linear programming. These problems will now be defined in appropriate generality.

a. Solution of linear equations.

Let X, Y denote complete normed linear spaces over a common scalar field Λ . In most applications, one has $\Lambda = \mathbb{R}$, the real numbers, or $\Lambda = \mathbb{C}$, the complex numbers. The notation $L(X, Y)$ will be used for the set of continuous linear operators from X into Y . Given an operator $A \in L(X, Y)$ and a vector $y \in Y$ as data, the problem is to find a solution $x \in X$ of the linear equation

$$(1.1) \quad Ax = y.$$

For practical as well as abstract treatment of this problem, it is important to be in possession of a *theory* of equation (1.1), which provides information as to which of the following alternatives holds:

Sponsored by the United States Army under Contract No.: DAAG29-75-C-0024.

$$(1.1a) \quad \left\{ \begin{array}{l} \text{(i)} \quad \text{For each } y \in Y, \text{ equation (1.1) has a unique solution } x \in X; \\ \text{or} \\ \text{(ii)} \quad \text{for some } y \in Y, \text{ equation (1.1) has no solution or several} \\ \text{solutions.} \end{array} \right.$$

The choice between (i) existence and uniqueness or (ii) nonexistence or nonuniqueness or solutions exhausts the logical possibilities, and thus the *alternative structure* (1.1a) will be characteristic of the theory of any equation, linear or nonlinear. In case (i), the operator A is said to be *nonsingular*; otherwise (case (ii)), it is called a *singular operator*.

b. Inversion of linear operators.

This problem is closely related to the solution of linear equations. In the nonsingular case (i), equation (1.1) defines the linear (right) *inverse operator* A^{-1} which gives the unique solution x as

$$(1.2) \quad x = A^{-1}y.$$

In many applications, one has $Y = X$ and A^{-1} satisfies

$$(1.3) \quad A^{-1}A = AA^{-1} = I,$$

where I denotes the *identity operator* in X , that is, $Ix = x$ for all $x \in X$.

In the singular case (ii), the alternatives are nonexistence or nonuniqueness of solutions x of (1.1). The inverse operator A^{-1} of A does not exist in this case, but one may seek a *generalized inverse* A^+ of A which has some properties which are desirable for the application at hand. For example, in connection with the problem of solving the linear equation (1.1), one might want

$$(1.4) \quad x = A^+y$$

to be a solution if the equation is *consistent*, and thus is satisfied by one or more elements of X . It turns out that this is equivalent to the condition that A^+ satisfies the operator equation

$$(1) \quad AA^+A = A.$$

Any operator A^+ for which (1) holds will be called an *inner inverse* of A [25, pp. 7-11]. The more formal term *{1}-inverse* of A [3, pp. 7-8] has also been applied specifically to operators A^+ satisfying condition (1).

If equation (1.1) is consistent and A^+ is an inner inverse of A , then all solutions x may be represented in the form

$$(1.5) \quad x = A^+y + (I - A^+A)z$$

for $z \in X$. With z arbitrary, formula (1.5) is called the *general solution* of equation (1.1), as in the elementary theory of linear differential equations.

Generalized inverses may also be useful in case equation (1.1) has no solutions, and thus is said to be *inconsistent*, or *overdetermined*. Here, the possibility of choosing a *generalized solution* x to minimize the norm of the *residual vector*

$$(1.6) \quad r = Ax - y$$

in Y is considered. Suppose that the set

$$(1.7) \quad G(A, y) = \{x \mid \|Ax - y\| = \min_{z \in X} \|Az - y\|\}$$

of generalized solutions x is nonempty, as will certainly be the case if equation (1.1) is consistent. If Y is a Hilbert space, then elements $x \in G(A, y)$ are ordinarily called *least-squares solutions* of the linear equation (1.1). Thus, one might require that $A^+y \in G(A, y)$ for all $y \in Y$ in addition to property (1). Furthermore, the subset

$$(1.8) \quad S(A, y) = \{x \mid x \in G(A, y), \|x\| = \min_{z \in G(A, y)} \|z\|\}$$

of $G(A, y)$ may be nonempty, and would then consist of the generalized (or least-squares) solutions x of (1.1) of *minimum norm*. The requirement that $A^+y \in S(A, y)$ for all $y \in Y$ would then also be a possible additional restriction on the set of generalized inverses of A . If $S(A, y)$ consists of a single point for each $y \in Y$, then the corresponding generalized inverse A^+ is uniquely determined. In case X and Y are finite-dimensional Euclidean spaces, this generalized inverse A^+ exists and is the *Moore-Penrose inverse* of A [3, pp. 7, 103-121], which, in addition to (1), satisfies the condition

$$(2) \quad A^+AA^+ = A^+;$$

that is, A^+ is also an *outer inverse* of A [25, pp. 12-14], and the symmetry conditions

$$(3) \quad (AA^+)^* = AA^+,$$

and

$$(4) \quad (A^+A)^* = A^+A,$$

where M^* denotes the conjugate transpose of the matrix M .

The problem of finding generalized solutions can become delicate in more general spaces, as the set $S(A, y)$ may consist of more than one element or be empty [20, 25]; in fact, $G(A, y)$ will be empty if the infimum of the norm of the residual vector is not attained. Of course, there are also many applications of generalized inverses in addition to the solution of linear equations in the singular case [3, 21], and this fairly recent subject already has a vast literature [24].

c. The eigenvalue-eigenvector problem.

This problem is posed most naturally in the case $Y = X$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$. One looks for scalars (real or complex numbers) λ and vectors $x \neq 0$ such that

$$(1.9) \quad Ax = \lambda x.$$

Solutions λ of this problem are called *eigenvalues* of the linear operator A ; for each eigenvalue λ , nonzero solutions x of (1.9) are said to be the corresponding *eigenvectors* of A . As equation (1.9) is homogeneous in x , the condition $x \neq 0$ may be replaced, for example, by

$$(1.10) \quad \langle x, x \rangle = 1,$$

or some other normalization condition.

From a standpoint of functional analysis, the determination of the eigenvalues of A is a special case of the more general problem of finding the *spectrum* $\sigma(A)$ of A . In a complex Hilbert space X , the set

$$(1.11) \quad \rho(A) = \{ \lambda \mid (A - \lambda I)^{-1} \in L(X, X) \}$$

of complex numbers λ is called the *resolvent* of A . Thus, $\lambda \in \rho(A)$ if and only if the operator $A - \lambda I$ has a continuous inverse. The spectrum of A is simply the complement of the resolvent,

$$(1.12) \quad \sigma(A) = C - \rho(A),$$

and hence contains any eigenvalues of A .

d. Linear programming.

In order to formulate this problem, suppose that X, Y are real spaces with partial ordering relationships denoted by \leq . For most applications, X and Y are taken to be finite-dimensional, in which case the partial ordering is the usual componentwise comparison of vectors [26, pp. 155-158]. Also needed is the *dual space* $X^* = L(X, R)$ of X ; that is, the space of continuous *linear functionals* defined on X . It is convenient to use the *bracket notation* of Dirac [7, pp. 18-28] for linear functionals. If $c \in X^*$, then define

$$(1.13) \quad \langle c, x \rangle := c(x),$$

which will be consistent with the notation for the inner product if X is a Hilbert space [7, pp. 6-8].

One formulation of the (primal) linear programming problem [26, pp. 156-157] is, given $A \in L(X, Y)$, $y \in Y$, $c \in X^*$, find $x \in X$ to maximize

$$(1.14) \quad f(x) = \langle c, x \rangle + \xi$$

subject to

$$(1.15) \quad Ax \leq y, \quad x \geq 0.$$

The function $f(x)$ defined by (1.14) is called the *objective function* of the problem, and conditions (1.15) are known as *constraints*.

Instead of the primal problem (1.14)-(1.15), one may wish to consider the *dual problem* [26, pp. 188-190] which is to find $z \in Y^*$ to minimize

$$(1.16) \quad g(z) = \langle z, y \rangle - \zeta$$

subject to

$$(1.17) \quad A^* z \geq c, \quad z \geq 0.$$

In (1.17), the operator $A^* \in L(Y^*, X^*)$ is the *adjoint* of A , defined by

$$(1.18) \quad \langle A^* y^*, x \rangle = \langle y^*, Ax \rangle$$

for all $y^* \in Y^*, x \in X$. It will also be convenient to write

$$(1.19) \quad y^* A := A^* y^*, \quad y^* \in Y^*;$$

that is, $y^* A$ is the linear functional on X defined by

$$(1.20) \quad (y^* A)x := y^*(Ax) = \langle y^*, Ax \rangle, \quad x \in X.$$

This is analogous to the notation frequently used in elementary matrix algebra, with x being considered to be a column vector, and y^* a row vector. The scalar quantity (1.20) will also be denoted by

$$(1.21) \quad \langle y^* Ax \rangle := \langle y^*, Ax \rangle.$$

The subject of perturbation methods and theory has a long history, and there is a vast literature devoted to this topic and its applications. The bibliography at the end of this paper, rather than attempting to be comprehensive, lists only references cited in the text, doubtless at the cost of omitting a number of significant contributions.

2. Perturbed linear problems. Perturbation theory, as applied to the linear problems listed in §1, starts from the assumption that their solutions are known for the given reference data $A \in L(X, Y)$, $y \in Y$, $c \in X^*$. The object is to study the behavior of these solutions for various classes of *perturbed data*.

$$(2.1) \quad B = A + \Delta A, \quad z = y + \Delta y, \quad d = c + \Delta c,$$

where the *perturbations* $\Delta A \in L(X, Y)$, $\Delta y \in Y$, $\Delta c \in X^*$ or appropriate information about them are given. One then desires to calculate or estimate the corresponding changes $\Delta x \in X$, $\Delta A^{-1} \in L(Y, X)$, $\Delta A^+ \in L(Y, X)$, $\Delta \lambda \in \Lambda$ in the solutions $x \in X$, $A^{-1} \in L(Y, X)$, $A^+ \in L(Y, X)$, $\lambda \in \Lambda$ of the original problems. Here

$$(2.2) \quad \Delta A^{-1} = B^{-1} - A^{-1}$$

denotes the difference between the inverse, if it exists, of the perturbed operator B and the inverse of the unperturbed operator A , and not $(\Delta A)^{-1}$, which may also exist. A similar observation applies to the notation ΔA^+ .

As an example, the perturbed linear system

$$(2.3) \quad Bw = z$$

can be solved for

$$(2.4) \quad w = x + \Delta x$$

if Δx can be obtained in terms of ΔA and Δy , the solution x of the unperturbed system (1.1) with reference data A, y being assumed to be known.

The motivation behind perturbation methods is that if the perturbations in the data are "small" in some sense, then one might expect the changes in the solutions to be correspondingly small, at least under suitable conditions. What is referred to here as "small" may vary widely, depending on the specific problem, the type of perturbation considered, the computing power available, and perhaps other factors. In the next section, a framework will be developed to characterize the concept of small perturbations more precisely.

The goals of perturbation theory may be either practical or theoretical. Two uses of perturbation methods in actual computation are to find solutions of perturbed problems with economy of effort, and to obtain error estimates. In the first case, computing the solution of a given linear problem might be extremely laborious, but a large amount of information could be generated in the process. One would then hope to be able to use this information to solve perturbations of the reference problem with less work than required when starting from scratch, as indicated in connection with the illustration (2.3)-(2.4) cited above. In the case of error estimation, the perturbations are considered to arise from inaccuracies in the data and from truncation and roundoff errors in the computation. Usually, these perturbations can only be estimated, and one seeks some kind of information about the possible error in the solution. One approach, called *forward error estimation*, starts from assumptions about the perturbations in the data, and obtains a comparison of the solution actually obtained with that of the reference problem if exact data and computation were employed. For *backward error estimation*, as developed by Wilkinson [34], the solution actually obtained is taken to be the exact solution of some perturbation of the reference problem, and estimates are made of the corresponding changes in the data. With the forward method, the computed solution is considered to be acceptable if it can be shown to be "close" to the (unknown) solution of the reference problem, while in the backward procedure, the criterion of acceptability is that the problem actually solved is "close" to the reference problem in some sense. More precise concepts of "closeness" will be introduced in the next section.

Perturbation methods can also be used for theoretical purposes. If a conceptual framework can be developed in which the problems considered can be viewed as perturbations of problems with known theory, then it may be possible to extend this theory

from one class to the other. This is the basis, for example, of the classical technique of Erhard Schmidt for obtaining the theory of linear Fredholm integral equations of second kind from the theory of finite linear algebraic systems [4, p. 155]. A more general situation will be described in a later section. Another theoretical use of perturbation methods, closely related to error estimation, is to determine the sensitivity of the solution of a linear problem to changes in the data. For example, one may wish to know which components of the solution are affected most strongly by a small change in one of the coefficients of the input data, and which are relatively undisturbed. This kind of analysis can also be used to pursue cause-and-effect relationships in mathematical models of various natural systems and processes.

3. Analytic and algebraic perturbations. For the present purposes, it will be convenient to classify perturbations into two nonexclusive categories; analytic and algebraic. This classification arises from the information available in each case and the methodology used to solve the perturbation problem, as well as an attempt to clarify what is meant by a "small" perturbation. In general, analytic perturbation theory uses metric information, and obtains solutions to perturbation problems in terms of series expansions, or by iterative methods. An objective criterion for a perturbation to be small in this case is that the required series or iterations converge. A more subjective condition is that the convergence be rapid enough to be useful in practice. The satisfaction of this restriction will depend, among other things, on the computing power available and whether the transformations involved can be carried out explicitly, or have to be approximated.

The idea of smallness for algebraic perturbations also depends more or less on outside factors. Here, the perturbations of operators are operators with finite-dimensional range, and vectors and functionals are perturbed by elements belonging to finite-dimensional subspaces of the corresponding spaces. The solution of algebraic perturbation problems will require solving finite algebraic problems of similar type, with the judgment as to what constitutes a "small" finite algebraic problem being again tied up with the resources available for computing. For example, early workers in the theory of linear integral equations knew that replacing them by a corresponding finite linear algebraic system would yield good approximate solutions, but despaired of being able to solve systems of order 10 or 20, as might be required to attain the desired accuracy [18, p. 242]. By contrast, today most computing centers are able to furnish the solutions of well-conditioned linear algebraic systems of order 100 or 200 at nominal cost.

More precise formulations will now be made of the type of information given and expected with each type of perturbation.

a. Analytic perturbations.

The fundamental metric information about vectors in Banach spaces X, Y, \dots is given, of course, by the respective norms $\| \cdot \|_X, \| \cdot \|_Y, \dots$. As confusion is unlikely, the subscripts will usually be dropped. If A is a continuous linear operator from X into Y , that is, if $A \in L(X, Y)$, then the numbers

$$(3.1) \quad M(A) = \sup_{\|x\|=1} \|Ax\|, \quad m(A) = \inf_{\|x\|=1} \|Ax\|,$$

exist and are finite [2, p. 54; 16, p. 194]. $M(A)$ and $m(A)$ are called the *upper* and *lower bound* of A , respectively. With the natural definitions of addition and scalar multiplication of linear operators, it is well known [38, p. 163] that $L(X, Y)$ is a Banach space for the *operator norm* $\|A\| = M(A)$. In some spaces, this norm is easy to compute, but in others, finding $M(A)$ might require more effort than solving the problem of interest. For numerical purposes, it is often convenient to assign a norm to the linear operator space $L(X, Y)$ which is easier to compute than the operator norm, and is *consistent* with it in the sense that

$$(3.2) \quad \|A\| \geq M(A).$$

For example, if $X = Y = E^n$, (complex) n -dimensional Euclidean space, then $A = (a_{ij})$ is represented by an $n \times n$ matrix with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$. One has

$$(3.3) \quad M(A) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_n|\},$$

which requires finding the eigenvalue of largest modulus of A . On the other hand, the *Euclidean norm* of A ,

$$(3.4) \quad \|A\| = \left(\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}}$$

is consistent and may be found by a straightforward calculation.

In the case of the adjoint spaces X^*, Y^*, \dots of continuous linear functionals on X, Y, \dots , the norm will always be defined analogously to the operator norm, that is,

$$(3.5) \quad \|c\| = \sup_{\|x\|=1} | \langle c, x \rangle |$$

for $c \in X^*$.

Thus, in the perturbed linear system (2.3), one would want a convergent process to calculate Δx , or an estimate for $\|\Delta x\|$ in terms of bounds for $\|\Delta A\|$ and $\|\Delta y\|$, and perhaps also the known quantities $\|A\|, \|x\|, \|y\|$.

b. Algebraic perturbations.

An algebraic perturbation Δy of a vector $y \in Y$ is defined to be an element of a finite-dimensional subspace

$$(3.6) \quad Y_n = \text{span}\{y_1, y_2, \dots, y_n\}$$

of Y consisting of all linear combinations of given independent basis vectors y_1, y_2, \dots, y_n in Y . A similar definition applies to algebraic perturbations of linear functionals. Ordinarily, algebraic perturbations will be restricted to subspaces with small dimension (in the sense described above). However, if the original spaces are finite-dimensional, it is of course possible to represent an arbitrary perturbation as an algebraic perturbation.

In the case of linear operators, algebraic perturbations are represented by linear operators with finite-dimensional ranges. Such operators are said to be of *finite rank*, or degenerate (in infinite-dimensional spaces). Here, the *dyadic* notation of Dirac [7, pp. 26-28] will be adopted; for $u \in Y$, $v \in X^*$, the symbol $u > < v$ will represent an operator of rank one from X into Y , with

$$(3.7) \quad (u > < v)x = u < v, x > = < v, x > u \in Y$$

for $x \in X$. Also, for $y^* \in Y^*$, the transposed operation will be denoted by

$$(3.8) \quad y^* (u > < v) = < y^*, u > v \in X^*,$$

again consistent with the notation introduced in §1. In these terms, a general algebraic perturbation $\Delta A \in L(X, Y)$ of rank n will be written as

$$(3.9) \quad \Delta A = \sum_{i=1}^n u_i > < v_i,$$

where the vectors $u_i \in Y$ and functionals $v_i \in X^*$, $i = 1, 2, \dots, n$, form linearly independent sets. The range of the operator (3.9) is $Y_n = \text{span}\{u_1, u_2, \dots, u_n\}$. In the finite-dimensional case, Y_n could coincide with Y , and arbitrary perturbations of linear operators could be written in the form (3.9).

Algebraic perturbations of vectors and linear operators are sometimes referred to as *finite rank modifications*. This terminology is useful if a clear distinction between analytic and algebraic methods is intended. By the use of algebraic perturbation theory, one would expect to obtain the perturbations in solutions of linear problems in the same form as the perturbations in the data. For example, one would want to express Δx as a linear combination of vectors x_1, x_2, \dots, x_n to be determined, that is, $\Delta x \in \text{span}\{x_1, x_2, \dots, x_n\} = X_n$, a finite-dimensional subspace of X . Similarly, expressions of the form (3.9) for ΔA^{-1} and ΔA^\dagger would be sought. In other words, algebraic perturbations in the data of linear problems are expected to give rise to finite rank modifications of their solutions.

In contrast to analytic perturbation theory, the use of algebraic methods does not involve restrictions on the norms of the perturbations in the data. However, it is possible that algebraic perturbations can be small in the analytic sense, so that either technique could be employed. Also, as illustrated in the next section, certain problems lend themselves to a combination of algebraic and analytic methods.

4. Compact operators and the Fredholm theory. A theoretical application of perturbation methods, which also has implications for numerical computation, is the extension of the theory of finite linear algebraic systems of n equations in n unknowns to certain types of linear equations (1.1) in infinite-dimensional spaces. An extension of this kind will be obtained here by the use of both analytic and algebraic techniques. First, the alternative structure (1.1a) of the theory of equation (1.1) will be given an explicit formulation for the class of operators to be

Definition 4.1. Linear operators belonging to a class $\mathcal{A} \subset L(X, Y)$ are said to have a *Fredholm theory* if for each $A \in \mathcal{A}$, either (i) the *homogeneous equation*

$$(4.1) \quad Ax = 0$$

has the unique solution $x = 0$, in which case the inhomogeneous equation (1.1) has a unique solution x for each $y \in Y$, or (ii) equation (4.1) has nonzero solutions, each of which can be expressed as a linear combination of a finite number d linearly independent solutions $x_1, x_2, \dots, x_d \in X$, in which case the *transposed homogeneous equation*

$$(4.2) \quad zA = 0$$

likewise has d linearly independent solutions $z_1, z_2, \dots, z_d \in Y^*$, in terms of which all its nonzero solutions are expressible as linear combinations, and the inhomogeneous equation (1.1) has no solutions unless

$$(4.3) \quad \langle z_i, y \rangle = 0, \quad i = 1, 2, \dots, d.$$

If (4.3) is satisfied and x_0 is any solution of (1.1) (sometimes called a *particular solution*), then the *general solution* of the inhomogeneous equation can be written as

$$(4.4) \quad x = x_0 + \sum_{i=1}^d \alpha_i x_i,$$

with arbitrary scalars $\alpha_1, \alpha_2, \dots, \alpha_d$.

For the algebraic case $X = Y = \mathbb{R}^n$, real n -dimensional space, the class \mathcal{A} of linear operators with Fredholm theory consists of all $n \times n$ real matrices $A = (a_{ij})$, that is, $\mathcal{A} = L(\mathbb{R}^n, \mathbb{R}^n)$, and the alternatives in Definition 4.1 were known to hold long before 1903, when the Norwegian mathematician Ivar Fredholm [6] established the correspondence between the theories of finite linear algebraic systems and linear integral equations of the form

$$(4.5) \quad x(s) - \lambda \int_0^1 K(s, t) x(t) dt = y(s), \quad 0 \leq s \leq 1,$$

giving rise to the present name for the theory.

Definition 4.2. A linear operator $K \in L(X, Y)$ is said to be *compact* if, given any $\epsilon > 0$, there exists a positive integer $n = n(\epsilon)$ such that

$$(4.6) \quad K = S + F,$$

where $\|S\| < \epsilon$ and F is of finite rank n .

A compact operator may thus be regarded as a small analytic perturbation of an operator of finite rank, or as a finite rank modification of an operator which is small in the analytic sense. It will be shown that the Fredholm theory can be extended to operators which can be expressed as the sum of a linear operator having a continuous inverse and a compact operator. That is, if $\mathcal{J} \subset L(X, Y)$ denotes the class of linear operators J such that $J^{-1} \in L(Y, X)$ exists, $\mathcal{K} \subset L(X, Y)$ the class of compact operators, and $\mathcal{A} = \mathcal{J} \oplus \mathcal{K}$ the class of linear operators of the form

$$(4.7) \quad A = J + K, \quad J \in \mathcal{J}, \quad K \in \mathcal{K}$$

then each $A \in \mathcal{A}$ has a Fredholm theory. This assertion will be proved in the next section by combining results from both analytic and algebraic perturbation theory. First, it will be shown that if $J \in \mathcal{J}$, then one has the well known result that $J + \Delta J \in \mathcal{J}$ for $\|\Delta J\|$ sufficiently small. Later, the Fredholm alternative given in Definition 4.1 will be established for operators which are the sum of invertible linear operators and linear operators of finite rank. The statement that operators of the form (4.7) have a Fredholm theory will then follow from Definition 4.2.

5. Nonsingular linear equations and operators. In this section, the problems of solving linear systems and the inversion of linear operators will be considered for the nonsingular case. Here, alternative (1.1a(i)) holds, and the inverse A^{-1} of the operator A exists.

a. Analytic perturbation of well-posed problems.

Definition 5.1. A problem is said to be *well-posed* if it has a unique solution which depends continuously on the data.

As a general rule, analytic perturbation methods are only successful when applied to well-posed problems. This can require the imposition of additional conditions on the data to insure uniqueness and continuous dependence of the solution, at least in some neighborhood of the solution of the reference problem. For the linear problems considered in this section to be well-posed, the continuity (and hence boundedness) of A^{-1} is required in addition to its existence. Consequently, it will be assumed that $A^{-1} \in L(Y, X)$ in the following discussion of the application of analytic perturbation theory. If A maps X onto Y , then it is well known that $A^{-1} \in L(Y, X)$ if and only if $m(A) > 0$ [2, pp. 145-150]. Lonseth [16, p. 194] has derived the relationship

$$(5.1) \quad m(A)M(A^{-1}) = M(A)m(A^{-1}) = 1$$

between the upper and lower bounds of a linear operator A with the continuous inverse A^{-1} . Furthermore, $(A + \Delta A)^{-1}$ exists if $M(\Delta A) < m(A)$. Using (5.1), this result may be stated in terms of consistent norms.

Theorem 5.1. If $\|\Delta A\| < \frac{1}{\|A^{-1}\|}$, then $(A+\Delta A)^{-1}$ exists and is given by

$$(5.2) \quad (A+\Delta A)^{-1} = \sum_{n=0}^{\infty} (-A^{-1}\Delta A)^n A^{-1}.$$

Proof: The hypothesis guarantees the convergence of the *Neumann series* on the right side of (5.2). Denoting this series by S , one finds by direct manipulation that $(A+\Delta A)S = I_Y$, the identity operator in Y , and $S(A+\Delta A) = I_X$; hence, $S = (A+\Delta A)^{-1}$. QED

Although the Neumann series expansion (5.2) is useful for theoretical purposes, it is likely to be too slowly convergent for practical computation. The partial sums

$$(5.3) \quad S_k = \sum_{n=0}^k (-A^{-1}\Delta A)^n A^{-1}$$

of the Neumann series (5.2) may be obtained by the simple iteration

$$(5.4) \quad S_0 = A^{-1}, \quad S_k = S_0 - (A^{-1}\Delta A)S_{k-1}, \quad k = 1, 2, \dots$$

From (5.2), for $\theta = \|A^{-1}\Delta A\|$,

$$(5.5) \quad \|(A+\Delta A)^{-1} - S_k\| \leq \frac{\theta^{k+1}}{1-\theta} \|A^{-1}\|.$$

In order to find a more efficient method, the *Hotelling-Lonseth algorithm* [17] may be adapted to this purpose. In this special case, the iteration process is

$$(5.6) \quad B_0 = A^{-1}, \quad B_k = [1 + (-A^{-1}\Delta A)^{2^{k+1}}] B_{k-1}, \quad k = 1, 2, \dots$$

It is easy to show by mathematical induction that $B_k = S_{2^k-1}$; hence, from (5.5),

$$(5.7) \quad \|(A+\Delta A)^{-1} - B_k\| \leq \frac{\theta^{2^k}}{1-\theta} \|A^{-1}\|,$$

so that the sequence $\{B_k\}$ defined by (5.6) converges quadratically to $(A+\Delta A)^{-1}$.

The only additional labor required over the more slowly convergent algorithm (5.4) is the repeated squaring of the small operator $-A^{-1}\Delta A$.

Attention will now be devoted to the estimation of the perturbations ΔA^{-1} and Δx in the inverse of the perturbed operator and the solution of the perturbed linear equation (2.3), respectively [14, 15, 16]. It will be helpful to introduce the notion of the *condition number* of a bounded linear operator. For $A \in L(X, Y)$, the exact condition number $\kappa(A)$ of A is defined to be

$$(5.8) \quad \kappa(A) = \frac{M(A)}{m(A)},$$

and is a measure of the distortion of the image in Y of the unit ball in X as transformed by the operator A . If A has a continuous inverse, then $\kappa(A) = M(A)M(A^{-1})$ by (5.1). For computational purposes, it may be expedient to use

consistent norms for $L(X,Y)$ and $L(Y,X)$, and the approximate condition number

$$(5.9) \quad k(A) = \|A\| \cdot \|A^{-1}\|,$$

which is an upper bound for $\kappa(A)$. The inequalities given below will be stated in terms of consistent norms and approximate condition numbers, but remain valid if these upper bounds are replaced by their exact values.

First, from (5.2),

$$(5.10) \quad \Delta A^{-1} = (A + \Delta A)^{-1} - A^{-1} = \sum_{n=1}^{\infty} (-A^{-1} \Delta A)^n A^{-1},$$

and thus,

$$(5.11) \quad \|\Delta A^{-1}\| \leq \frac{\|A^{-1}\|^2 \|\Delta A\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|}.$$

Dividing (5.11) by $\|A^{-1}\|$ and multiplying and dividing $\|\Delta A\|$ on the right hand side by $\|A\|$ gives

$$(5.12) \quad \frac{\|\Delta A^{-1}\|}{\|A^{-1}\|} \leq \frac{k(A) \frac{\|\Delta A\|}{\|A\|}}{1 - k(A) \frac{\|\Delta A\|}{\|A\|}},$$

which expresses the relative change in the inverse in terms of the relative perturbation of the reference operator and its (approximate) condition number. A similar expression will now be obtained for the perturbation Δx in the solution of (2.3).

Theorem 5.2. If $\|\Delta A\| < \frac{1}{\|A^{-1}\|}$, then the perturbed linear equation (2.3)

has a unique solution $w = x + \Delta x$ for each $z = y + \Delta y$, and

$$(5.13) \quad \frac{\|\Delta x\|}{\|x\|} \leq \frac{k(A)}{1 - k(A) \frac{\|\Delta A\|}{\|A\|}} \left[\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta y\|}{\|y\|} \right]$$

provided, of course, that $y \neq 0$.

Proof: By Theorem 5.1, the hypothesis guarantees that $B^{-1} = (A + \Delta A)^{-1} = A^{-1} + \Delta A^{-1}$ exists, which implies the unique solvability of (2.3) for each z . Writing (2.3) as

$$(5.14) \quad (A + \Delta A)(x + \Delta x) = y + \Delta y,$$

one obtains

$$(5.15) \quad \Delta x = \Delta A^{-1} y + (A + \Delta A)^{-1} \Delta y.$$

As $y = Ax$, from (5.10),

$$(5.16) \quad \Delta A^{-1} y = \sum_{n=1}^{\infty} (-A^{-1} \Delta A)^n x,$$

so that

$$(5.17) \quad \|\Delta A^{-1}y\| \leq \frac{\|A^{-1}\| \cdot \|\Delta A\| \cdot \|x\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} = \frac{k(A)}{1 - k(A) \frac{\|\Delta A\|}{\|A\|}} \cdot \frac{\|\Delta A\|}{\|A\|} \cdot \|x\|$$

Similarly, from (5.2) and the fact that $\|y\| = \|Ax\| \leq \|A\| \cdot \|x\|$,

$$(5.18) \quad \|(A+\Delta A)^{-1}\Delta y\| \leq \frac{\|A^{-1}\| \cdot \|\Delta y\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \cdot \frac{\|A\| \cdot \|x\|}{\|y\|} =$$

$$\frac{k(A)}{1 - k(A) \frac{\|\Delta A\|}{\|A\|}} \cdot \frac{\|\Delta y\|}{\|y\|} \cdot \|x\|.$$

Inequality (5.13) now follows directly from (5.15), (5.17), and (5.18). QED

b. Algebraic perturbation of nonsingular linear equations and operators.

The simplest type of algebraic perturbation (2.3) of the linear system (1.1) is with $\Delta A = 0$ and Δy restricted to belong to a finite-dimensional subspace Y_n of Y . Given a basis $\{y_1, y_2, \dots, y_n\}$ for Y_n , one need only find the corresponding basis vectors

$$(5.19) \quad x_i = A^{-1}y_i, \quad i = 1, 2, \dots, n,$$

of the subspace $X_n \subset X$ which will then contain all possible perturbations Δx .

Thus, given

$$(5.20) \quad \Delta y = \alpha_1 y_1 + \alpha_2 y_2 + \dots + \alpha_n y_n,$$

it follows that

$$(5.21) \quad \Delta x = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n.$$

In actual computation, it may be more efficient to solve the n systems $Ax_i = y_i$, $i = 1, 2, \dots, n$, for the basis vectors for X_n , even if X is finite-dimensional [5, p. 77], than to calculate A^{-1} .

To introduce the study of the effect of a finite-rank modification of an operator upon its inverse, the case of rank one perturbation will be considered first, as all the indicated operations can be displayed explicitly. For $\Delta A = u \langle v$ with $u \in Y$, $v \in X^*$ nonzero, the solvability of the perturbed system (2.3), that is

$$(5.22) \quad (A + u \langle v)w = z,$$

will be investigated for arbitrary z . As A^{-1} is assumed to exist, the equations $A\hat{u} = u$, $A\hat{z} = z$ can be solved uniquely for $\hat{u} = A^{-1}u$, $\hat{z} = A^{-1}z$, respectively. In terms of these solutions, (5.22) may be written as

$$(5.23) \quad w = \hat{z} - \hat{u} \langle v, w \rangle.$$

The key to the solvability of (5.23), and hence of (5.22), is the determination of the number $\xi = \langle v, w \rangle$. From (5.23),

$$(5.24) \quad \langle v, w \rangle + \langle v, \hat{u} \rangle \langle v, w \rangle = \langle v, \hat{z} \rangle$$

If the determinant

$$(5.25) \quad \delta = 1 + \langle v, \hat{u} \rangle = 1 + \langle v A^{-1} u \rangle$$

does not vanish, then (5.24) has the unique solution

$$(5.26) \quad \langle v, w \rangle = \frac{\langle v, \hat{z} \rangle}{\delta} = \frac{\langle v A^{-1} z \rangle}{1 + \langle v A^{-1} u \rangle},$$

where the notation (1.21) has been used in (5.25) and (5.26). Substitution of (5.26) into (5.23) yields

$$(5.27) \quad w = A^{-1} z - A^{-1} u \frac{\langle v A^{-1} z \rangle}{\delta} = \left(A^{-1} - \frac{A^{-1} u \langle v A^{-1} \rangle}{1 + \langle v A^{-1} u \rangle} \right) z,$$

so that

$$(5.28) \quad (A + u \langle v \rangle)^{-1} = A^{-1} - \frac{A^{-1} u \langle v A^{-1} \rangle}{1 + \langle v A^{-1} u \rangle},$$

provided $\delta \neq 0$. Hence, the inverse of a rank one modification of an invertible operator, if it exists, is a rank one modification of the inverse of the reference operator. The symmetry of (5.28), sometimes called the *Sherman-Morrison-Woodbury formula* [11, pp. 123-124; 35, 46], is appealing.

Using (5.28), the solution $w = x + \Delta x$ of (5.22) is

$$(5.29) \quad x + \Delta x = x + A^{-1} \Delta y - \frac{\langle v, x + A^{-1} \Delta y \rangle}{1 + \langle v A^{-1} u \rangle} A^{-1} u,$$

or

$$(5.30) \quad \Delta x = A^{-1} \Delta y - \frac{\langle v, x + A^{-1} \Delta y \rangle}{1 + \langle v A^{-1} u \rangle} A^{-1} u.$$

Thus, the perturbation Δx is a linear combination of $A^{-1} \Delta y$ and the vector $\hat{u} = A^{-1} u$. If Δy is an algebraic perturbation of the form (5.20), and \hat{u} is independent of the vectors $x_i = A^{-1} y_i$, $i = 1, 2, \dots, n$, then Δx will lie in the $(n+1)$ -dimensional subspace $X_{n+1} = \text{span} \{\hat{u}, x_1, x_2, \dots, x_n\}$ of X ; otherwise $\Delta x \in X_n = \text{span} \{x_1, x_2, \dots, x_n\}$.

Before going to the general case, two applications of algebraic perturbation theory will be given which involve rank one modifications. The first is to the Fredholm integral equation (4.5) in which the kernel $K(s, t)$ has the special form

$$(5.31) \quad K(s, t) = \begin{cases} u(t)v(s), & 0 \leq t \leq s \leq 1, \\ u(s)v(t), & 0 \leq s \leq t \leq 1. \end{cases}$$

This type of kernel arises in applications; for example, as a Green's function determined by a two-point boundary value problem [32]. Given the representation (5.31) for $K(s,t)$, the integral equation (4.5) may be written as

$$(5.32) \quad x(s) - \lambda \int_0^s L(s,t) x(t) dt - \lambda \int_0^1 u(s)v(t)x(t) dt = y(s),$$

where

$$(5.33) \quad L(s,t) = u(t)v(s) - u(s)v(t), \quad 0 \leq t \leq s \leq 1.$$

Equation (5.32) is of the form $(A - \lambda u > v)x = y$, where $A = I - \lambda L$ is a linear Volterra integral operator of second kind with kernel (5.33), and $u > v$ is a Fredholm integral operator of first kind and rank one with kernel $u(s)v(t)$. The inverse $A^{-1} = (I - \lambda L)^{-1}$ of the Volterra operator of second kind exists for all λ [30, pp. 52-53], and thus the linear Volterra integral equation

$$(5.34) \quad \hat{w}(s) - \lambda \int_0^s L(s,t) \hat{w}(t) dt = w(s)$$

can be solved for arbitrary $w(s)$; in particular, one obtains $\hat{w}(s) = \hat{u}(s)$ for $w(s) = u(s)$, and $\hat{w}(s) = \hat{y}(s)$ for $w(s) = y(s)$. Corresponding to (5.25), if the Fredholm determinant

$$(5.35) \quad \delta = 1 - \lambda \langle v, \hat{u} \rangle = 1 - \lambda \int_0^1 v(t) \hat{u}(t) dt$$

does not vanish, then, from (5.27),

$$(5.36) \quad x(s) = \hat{y}(s) + \frac{\lambda}{\delta} \hat{u}(s) \int_0^1 v(t) \hat{y}(t) dt$$

is the unique solution of (5.32). Hence, the solution of the Fredholm integral equation (4.5) with the kernel (5.31) can be obtained by solving the Volterra integral equation (5.34) with right-hand sides $w(s) = u(s)$ and $w(s) = y(s)$, followed by the calculation of the inner product integrals in (5.35) and (5.36).

The second application to be considered for rank one modification of a linear operator is to backward error analysis in the solution of linear equations. Suppose that one attempts to solve the linear equation (1.1) and obtains, instead of x , an approximate solution w such that

$$(5.37) \quad A w = y + r,$$

with nonzero residual r . The Hahn-Banach theorem [38, p. 186] guarantees the existence of a linear functional $w^* \in X^*$ such that $\|w^*\| = 1$ and $\langle w^*, w \rangle = \|w\|$. Thus, w is the exact solution of the linear equation

$$(5.38) \quad \left(A - \frac{r > w^*}{\|w\|} \right) w = y$$

with perturbed operator and desired right-hand side. An analytic bound for the perturbation of A is thus

$$(5.39) \quad \|\Delta A\| = \frac{\|r > w^*\|}{\|w\|} = \frac{\|r\|}{\|w\|}.$$

Returning to the study of general algebraic perturbations, note that the equivalence of (5.22) and the single scalar equation (5.24) establishes that (5.22) has a Fredholm theory, because (5.24) does. In the case $\delta = 0$, the homogeneous equation $(A + u > < v)w = 0$ is satisfied by (and only by) vectors $w = \alpha \hat{u}$ with α arbitrary. The inhomogeneous equations (5.22) and (5.24) then have solutions only if

$$(5.40) \quad \langle v, \hat{z} \rangle = \langle vA^{-1}z \rangle = \langle \hat{v}, z \rangle = 0,$$

where $\hat{v} = vA^{-1}$ satisfies the transposed homogeneous equation

$$(5.41) \quad \hat{v}(A + u > < v) = 0.$$

Theorem 5.3. If $\mathcal{J} \subset L(X, Y)$ denotes the class of all invertible linear operators, and $\mathcal{F} \subset L(X, Y)$ the class of all linear operators of finite rank, then all linear operators belonging to the class $\mathcal{Q} = \mathcal{J} * \mathcal{F}$ have a Fredholm theory.

Proof: If $B \in \mathcal{Q} = \mathcal{J} * \mathcal{F}$, then there is an invertible linear operator $A \in \mathcal{J}$ for which B can be written as

$$(5.42) \quad B = A + \sum_{j=1}^n u_j > < v_j,$$

where $u_j \in Y$, $v_j \in X^*$, $j = 1, 2, \dots, n$, are linearly independent sets of vectors and functionals, respectively. Equation (2.3) in this case is equivalent to

$$(5.43) \quad w = \hat{z} - \sum_{j=1}^n \hat{u}_j \langle v_j, w \rangle,$$

where $\hat{z} = A^{-1}z$, $\hat{u}_j = A^{-1}u_j$, $j = 1, 2, \dots, n$. Applying the functionals v_1, v_2, \dots, v_n to (5.43) in turn gives the equivalent finite linear algebraic system of equations

$$(5.44) \quad \xi_i + \sum_{j=1}^n \alpha_{ij} \xi_j = \zeta_i, \quad i = 1, 2, \dots, n$$

for $\xi_i = \langle v_i, w \rangle$, where $\zeta_i = \langle v_i, \hat{z} \rangle$ and $\alpha_{ij} = \langle v_i, \hat{u}_j \rangle$, $i, j = 1, 2, \dots, n$.

As the Fredholm alternative applies to (5.44), it follows that operators of the form

(5.42) have a Fredholm theory. QED

In the nonsingular case, an expression can be obtained for B^{-1} as a finite rank modification of A^{-1} . Let

$$(5.45) \quad M = (\delta_{ij} + \alpha_{ij})$$

denote the matrix of coefficients of the linear system (5.44), where δ_{ij} is the Kronecker delta: $\delta_{ij} = 0$ if $i \neq j$, $\delta_{ii} = 1$. As the determinant δ of M is assumed to be nonzero, the inverse of M may be written

$$(5.46) \quad M^{-1} = \frac{1}{\delta} (\beta_{ij}),$$

and thus

$$(5.47) \quad \xi_i = \langle v_i, w \rangle = \frac{1}{\delta} \sum_{j=1}^n \beta_{ij} \zeta_j = \frac{1}{\delta} \sum_{j=1}^n \beta_{ij} \langle v_j, \hat{z} \rangle,$$

$i = 1, 2, \dots, n$. Using (5.43) and the fact that $\langle v_j, \hat{z} \rangle = \langle v_j A^{-1}z \rangle = \langle \hat{v}_j, z \rangle$ for

$\hat{v}_j = v_j A^{-1}$, $i = 1, 2, \dots, n$, one obtains the solution w of (2.3) in this case as

$$(5.48) \quad w = \left(A^{-1} - \frac{1}{\delta} \sum_{i=1}^n \sum_{j=1}^n A^{-1} u_i > \beta_{ij} < v_j A^{-1} \right) z.$$

By taking appropriate linear combinations $\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_n$ of u_1, u_2, \dots, u_n and $\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_n$ of v_1, v_2, \dots, v_n (for example, by an *LU-decomposition* of M^{-1} [5, pp. 27-32]), (5.48) may be put in the form

$$(5.49) \quad w = \left(A^{-1} - \frac{1}{\delta} \sum_{j=1}^n A^{-1} \tilde{u}_j > \tilde{v}_j A^{-1} \right) z,$$

from which

$$(5.50) \quad \left(A + \sum_{j=1}^n u_j > \tilde{v}_j \right)^{-1} = A^{-1} - \frac{1}{\delta} \sum_{j=1}^n A^{-1} \tilde{u}_j > \tilde{v}_j A^{-1},$$

which is analogous to (5.28).

Another way to find the inverse of the perturbed operator (5.42) is the method of successive rank one modifications, which does not require obtaining M^{-1} explicitly. Set

$$(5.51) \quad B_0 = A, \quad B_0^{-1} = A^{-1},$$

and then the algorithm

$$(5.52) \quad \begin{cases} B_k = B_{k-1} + u_k > v_k, \\ B_k^{-1} = B_{k-1}^{-1} - \frac{B_{k-1}^{-1} u_k > v_k B_{k-1}^{-1}}{1 + < v_k B_{k-1}^{-1} u_k >}, \end{cases}$$

$k = 1, 2, \dots, n$ will give $B^{-1} = B_n^{-1}$ if none of the intermediate determinants

$$(5.53) \quad \delta_k = 1 + < v_k B_{k-1}^{-1} u_k >, \quad k = 1, 2, \dots, n,$$

vanish.

It should be noted again that it is not necessary to obtain A^{-1} to solve the equation (2.3) for

$$(5.54) \quad w = \hat{z} - \sum_{j=1}^n \xi_j \hat{u}_j,$$

as given by (5.43). What is required is to solve equation (1.1) for the $n+1$ right-hand sides $y = z, u_1, u_2, \dots, u_n$ for $x = \hat{z}, \hat{u}_1, \hat{u}_2, \dots, \hat{u}_n$, calculate the coefficients of the system (5.44) of n equations for the n unknowns $\xi_1, \xi_2, \dots, \xi_n$, solve this system, and then form the linear combination (5.54).

An important application of the above technique of algebraic perturbation is to the numerical solution of partial differential equations by what is called the *capacitance matrix method* [36, 42]. The basic problem is to solve, for example, the Poisson or Helmholtz equation on a region Ω , with information given on its boundary $\partial\Omega$ (see Figure 5.1). The use of finite-difference methods will lead to a linear

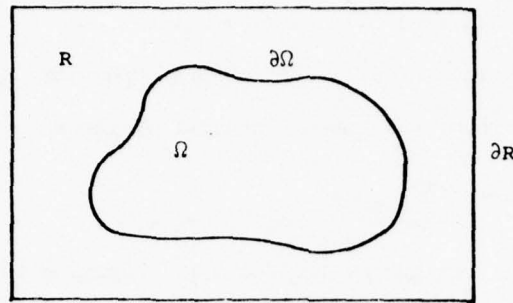


Figure 5.1

algebraic system $Bw = z$ which may be very laborious to solve. On the other hand, rapid and effective methods may be available for the algebraic system $Ax = y$ corresponding to the same finite-difference approximation to the problem posed on an enclosing rectangle R with boundary ∂R . By regarding the algebraic system obtained for Ω as a finite rank perturbation of the easily solved system arising from the approximate problem on R , a considerable reduction in effort may be possible. Typically, if the order of the systems (1.1) and (2.3) is about n^2 , then the rank of the perturbations ΔA and Δy will be approximately n .

The Fredholm theory will now be shown to apply to operators which are the sum of a continuously invertible operator and a compact operator.

Theorem 5.4. Operators A belonging to the class \mathcal{A} defined by (4.7) have a Fredholm theory.

Proof: Choose $\epsilon < 1/\|J^{-1}\|$. According to Definition 4.2, the compact operator K may be written as

$$(5.55) \quad K = S + \sum_{j=1}^n u_j \langle \cdot, v_j \rangle,$$

where $n = n(\epsilon)$ is finite. Thus,

$$(5.56) \quad A = J + S + \sum_{j=1}^n u_j \langle \cdot, v_j \rangle,$$

and Theorem 5.1 guarantees the existence of the inverse operator $(J+S)^{-1} \in L(Y, X)$.

It follows from Theorem 5.3 that A has a Fredholm theory. QED

Theorem 5.4 provides a basis for the "kernel splitting" method due to Erhard Schmidt [4, p. 155] for proving the Fredholm Alternative Theorem [6] for the linear integral equation (4.5). Suppose that $K(s, t)$ is continuous, or at least can be approximated sufficiently well by a kernel of finite rank so that one can write

$$(5.57) \quad K(s,t) = S(s,t) + \sum_{j=1}^n u_j(s)v_j(t)$$

where $S(s,t)$ is the kernel of a linear integral operator S with $\|S\| < \frac{1}{|\lambda|}$ in the appropriate norm. Then, the linear integral operator $I - \lambda K$ in (4.5) has the form

$$(5.58) \quad I - \lambda K = I - \lambda S - \lambda \sum_{j=1}^n u_j \langle \cdot, v_j \rangle,$$

where $(I - \lambda S)^{-1} = T(\lambda)$ exists by Theorem 5.1. Applying this operator to the equation $(I - \lambda K)x = y$, the integral equation (4.5) is seen to be equivalent to the linear equation

$$(5.59) \quad (I - \lambda \sum_{j=1}^n T(\lambda)u_j \langle \cdot, v_j \rangle)x = T(\lambda)y,$$

and thus $I - \lambda K$ has a Fredholm theory by Theorem 5.3. This approach regards $I - \lambda K$ as an algebraic perturbation of the invertible operator $I - \lambda S$.

On the other hand, suppose that

$$(5.60) \quad (I - \lambda \sum_{j=1}^n u_j \langle \cdot, v_j \rangle)^{-1} = I + \frac{\lambda}{\delta} \sum_{j=1}^n \tilde{u}_j \langle \cdot, \tilde{v}_j \rangle = Z(\lambda)$$

exists, where the notation (5.50) has been used. Then, $I - \lambda K$ is an analytic perturbation of an invertible operator, and (4.5) is equivalent to the equation

$$(5.61) \quad (I - \lambda Z(\lambda)S)x = Z(\lambda)y.$$

From Theorem 5.1, if $(I - \lambda K)^{-1}$ exists and $\|\lambda S\| < 1/\|(I - \lambda K)^{-1}\|$, then $Z(\lambda)$ exists, so all sufficiently good finite rank approximations (5.57) to $K(s,t)$ will lead to a solvable perturbed equation

$$(5.62) \quad (I - \lambda \sum_{j=1}^n u_j \langle \cdot, v_j \rangle)w = y$$

which is equivalent to a finite linear algebraic system of the form (5.44). Conversely, if the inverse operator $Z(\lambda)$ exists and $\|\lambda Z(\lambda)S\| < 1$, then it follows from the same theorem that $(I - \lambda K)^{-1}$ exists. If u_1, u_2, \dots, u_n and v_1, v_2, \dots, v_n are chosen so that all the inner products required can be calculated explicitly, then this gives a method for concluding the existence and uniqueness of the solution of the integral equation (4.5) on the basis of a finite set of algebraic computations, as well as a technique to obtain approximate solutions. An error analysis for (5.62) may be carried out by the analytic methods of §5a with $\Delta A = -\lambda S$, $\Delta y = 0$. A similar approach can be used on (5.59) with $T(\lambda)$ replaced by

$$(5.63) \quad T_k(\lambda) = I + \lambda S + \lambda^2 S^2 + \dots + \lambda^k S^k.$$

Setting $z = T_k(\lambda)y$, the perturbed equation

$$(5.64) \quad (I - \lambda \sum_{j=1}^n T_k(\lambda)u_j \langle \cdot, v_j \rangle)w = z$$

may be analyzed by the same technique, with

$$(5.65) \quad \Delta A = \lambda \sum_{j=1}^n (\lambda S)^{k+1} T(\lambda) u_j > v_j,$$

and

$$(5.66) \quad \Delta T(\lambda) y = -(\lambda S)^{k+1} T(\lambda) y.$$

As $\|T(\lambda)\| \leq 1/(1 - \|\lambda S\|)$, and for

$$(5.67) \quad \lambda F = \lambda \sum_{j=1}^n u_j > v_j,$$

one has

$$(5.68) \quad \|\Delta A\| \leq \frac{\|\lambda S\|^{k+1}}{1 - \|\lambda S\|} \|\lambda F\|, \quad \|\Delta T(\lambda) y\| \leq \frac{\|\lambda S\|^{k+1}}{1 - \|\lambda S\|} \|y\|$$

with

$$(5.69) \quad \|\lambda F\| \leq |\lambda| \sum_{j=1}^n \|u_j\| \cdot \|v_j\|$$

from (5.67).

Another analytic approach to the approximate solution of (4.5) which also yields error bounds is to solve (5.61) by iteration, as $\|\lambda z(\lambda) S\| < 1$ [29].

6. The singular case and generalized inverses. Attention will now be devoted to linear problems which are ill-posed because the linear operator involved does not have a bounded inverse. As the solutions, if any, of ill-posed problems do not depend on the data in a continuous fashion, it might be expected in this situation that analytic perturbation methods will be of little utility, or can be applied only under very restrictive conditions. For example, there is an inherent limitation as to how well an operator $B \in L(X, Y)$ without a continuous inverse can be approximated by an operator A belonging to the class $\mathcal{G} \subset L(X, Y)$ of operators with continuous inverses $A^{-1} \in L(Y, X)$. From Theorem 5.1,

$$(6.1) \quad \|B - A\| = \|\Delta A\| \geq \frac{1}{\|A^{-1}\|},$$

otherwise, the assumption that $B \in \mathcal{G}$ would be contradicted. Also, from (6.1),

$$(6.2) \quad \|A^{-1}\| \geq \frac{1}{\|B - A\|} = \frac{1}{\|\Delta A\|},$$

so that $\|A^{-1}\|$ and the approximate condition number

$$(6.3) \quad k(A) \geq \frac{\|A\|}{\|\Delta A\|} \geq \left| \frac{\|B\|}{\|\Delta A\|} - 1 \right|$$

grow without limit as $\|\Delta A\| \rightarrow 0$. Clearly, computational difficulties can be expected in the calculation of A^{-1} or in the solution of the linear equation (1.1) if A is very close in the analytic sense to an operator B which does not have a continuous inverse.

Theorem 6.1. If $\{A_n\} \subset \mathcal{J}$ is any sequence of linear operators such that $\lim_{n \rightarrow \infty} \|A_n - B\| = 0$, then $B \notin \mathcal{J}$ if and only if (6.2) holds for each $A = A_n$, $n = 1, 2, \dots$.

Proof: If $B \notin \mathcal{J}$, then it has already been shown that (6.2) holds for each A_n . To show the converse, suppose that $B \in \mathcal{J}$, and choose n sufficiently large so that $\|\Delta A_n\| = \|A_n - B\| < 1/2 \|B^{-1}\|$. It then follows from (5.2) that

$$(6.4) \quad \|A_n^{-1}\| \leq \frac{\|B^{-1}\|}{1 - \|\Delta A_n\| \cdot \|B^{-1}\|} < \frac{1}{\|\Delta A_n\|},$$

a contradiction of (6.2) which proves the theorem. QED

An evident drawback of analytic perturbation theory is that, in general, no conclusions can be drawn from the existence of $A^{-1} \in L(Y, X)$ about the invertibility or noninvertibility of any operator B for which inequality (6.1) holds. The algebraic theory, on the other hand, states that if B is the finite rank modification (5.42) of an invertible linear operator $A \in \mathcal{J}$, then B^{-1} exists if and only if

$$(6.5) \quad \delta = \det(\delta_{ij} + \langle v_i A^{-1} u_j \rangle) \neq 0.$$

Of course, one would still expect computational difficulty if B is nearly singular, especially if the inner products $\alpha_{ij} = \langle v_i A^{-1} u_j \rangle$, $i, j = 1, 2, \dots, n$, can only be calculated approximately.

The algebraic approach also provides information in the singular case. Supposing that $\delta = 0$, consider the transposed homogeneous equation

$$(6.6) \quad t \left(A + \sum_{i=1}^n u_i \langle v_i \rangle \right) = 0$$

for $t \in X^*$. Using the technique of §5b, this is equivalent to the finite linear algebraic system

$$(6.7) \quad \tau_j + \sum_{i=1}^n \tau_i \alpha_{ij} = 0, \quad j = 1, 2, \dots, n,$$

for $\tau_j = \langle t, u_j \rangle$. The system of equations (6.7) is the transposed homogeneous system corresponding to (5.44). If $\delta = 0$, then (6.7) has d linearly independent solutions

$$(6.8) \quad \tau^{(k)} = (\tau_1^{(k)}, \tau_2^{(k)}, \dots, \tau_n^{(k)}), \quad k = 1, 2, \dots, d,$$

and, corresponding to these, equation (6.6) also has d linearly independent solutions

$$(6.9) \quad t^{(k)} = \sum_{i=1}^n \tau_i^{(k)} v_i A^{-1} = \sum_{i=1}^n \tau_i^{(k)} \hat{v}_i,$$

$k = 1, 2, \dots, d$. Likewise, the homogeneous system

$$(6.10) \quad \xi_i + \sum_{j=1}^n \alpha_{ij} \xi_j = 0, \quad i = 1, 2, \dots, n,$$

has d linearly independent solutions

$$(6.11) \quad \xi^{(k)} = (\xi_1^{(k)}, \xi_2^{(k)}, \dots, \xi_n^{(k)})^T, \quad k = 1, 2, \dots, d,$$

from which are obtained the corresponding linearly independent solutions

$$(6.12) \quad w^{(k)} = \sum_{j=1}^n \xi_j^{(k)} A^{-1} u_j = \sum_{j=1}^n \xi_j^{(k)} \hat{u}_j,$$

$k = 1, 2, \dots, d$, of the homogeneous equation

$$(6.13) \quad (A + \sum_{j=1}^n u_j \langle v_j |) w = 0.$$

Representing the right-hand sides of the system (5.44) as the vector

$$(6.14) \quad \zeta = (\zeta_1, \zeta_2, \dots, \zeta_n)^T = (\langle v_1 A^{-1} z \rangle, \langle v_2 A^{-1} z \rangle, \dots, \langle v_n A^{-1} z \rangle)^T,$$

it is seen immediately that the conditions for the solvability of the finite inhomogeneous system (5.44) and the equivalent inhomogeneous equation (2.3) for the case $\delta = 0$ are

$$(6.15) \quad \langle \tau^{(k)}, \zeta \rangle = \langle \sum_{i=1}^n \tau_i^{(k)} \hat{v}_i, z \rangle = \langle t^{(k)}, z \rangle = 0,$$

$k = 1, 2, \dots, d$; that is, z must be orthogonal to all solutions of the homogeneous equation (6.6). If (6.15) is satisfied, then the general solution of (2.3) may be written as

$$(6.16) \quad w = \hat{w} + \sum_{k=1}^d \alpha_k w^{(k)},$$

where \hat{w} is some particular solution of (2.3), and the complementary vectors

$$(6.17) \quad \tilde{w} = \tilde{w}(\alpha_1, \alpha_2, \dots, \alpha_k) = \sum_{k=1}^d \alpha_k w^{(k)}$$

satisfy the homogeneous equation (6.13) for arbitrary $\alpha_1, \alpha_2, \dots, \alpha_k$.

Usually, in actual computational solution of linear equations, the distinction between the singular and nonsingular cases is not as clear-cut as in the alternatives (1.1a) or the Fredholm theory. In practice, an objective or subjective standard is set for what constitutes an "acceptable" (approximate) solution, and one of the following situations is observed:

$$(6.18) \quad \left\{ \begin{array}{l} \text{(i) An acceptable solution is obtained,} \\ \text{or} \\ \text{(ii) either no solution at all is found, or the computed} \\ \text{solution is unacceptable.} \end{array} \right.$$

In the computationally singular case (6.18ii), the method used to solve (1.1) or invert A may break down because A does not have a bounded inverse, or is analytically close to an operator $B \notin \mathcal{J}$. On the other hand, the algorithm employed may actually be trying to solve the system (5.44) with $\delta = 0$ and without (6.15) holding to the desired degree of accuracy. This will be called an *algebraic catastrophe of type I*. In the second situation described in (6.18ii), the acceptable particular

solution \hat{w} may be contaminated by a complementary vector (6.17) to the extent that the resulting solution is unacceptable. This *algebraic catastrophe of type II* can occur in the numerical solution of differential equations by the use of approximating difference equations. For example, the difference equation

$$(6.19) \quad 3u_{n+1} + 8u_n - 3u_{n-1} = 0$$

with the initial conditions

$$(6.20) \quad u_0 = 1, \quad u_1 = \frac{1}{3},$$

has the bounded solutions

$$(6.21) \quad u_n = \left(\frac{1}{3}\right)^n, \quad n = 0, 1, 2, \dots,$$

which may be the ones considered to be acceptable. However, a slight perturbation of (6.20), such as rounding $\frac{1}{3}$ to eight decimal places,

$$(6.22) \quad w_0 = 1, \quad w_1 = 0.33333333$$

gives the corresponding solutions w_n of $3w_{n+1} + 8w_n - 3w_{n-1} = 0$ as

$$(6.23) \quad w_n = (0.99999999)\left(\frac{1}{3}\right)^n + (0.00000001)(-3)^n,$$

$n = 1, 2, \dots$, and the second term on the right-hand side of (6.23) will eventually wreak havoc with the accuracy of the approximation of u_n by w_n .

As indicated in §1b, if the operator A is singular, then a generalized inverse A^+ of A having certain useful properties may be sought, for example, to give a solution of (1.1) in the form (1.4) if (1.1) is consistent. As (1.5) indicates, the vector $x = A^+y$ will be a particular solution of (1.1) for any inner inverse A^+ of A . An algebraic perturbation method may be used to obtain inner inverses of singular operators which have a Fredholm theory, under the technical assumption that the space Y is reflexive, that is, $Y^{**} = Y$ [38, p.192]. In this case, if

$$(6.24) \quad U^* = \{u_1^*, u_2^*, \dots, u_d^*\} \subset Y^*$$

is a set of linearly independent functionals on Y , then the Hahn-Banach theorem guarantees the existence of a set of d linearly independent vectors in Y to which the Gram-Schmidt orthonormalization process [38, p. 116] may be applied, if necessary, to obtain the set

$$(6.25) \quad U = \{u_1, u_2, \dots, u_d\} \subset Y$$

for which

$$(6.26) \quad \langle u_i^*, u_j \rangle = \langle u_i^*, u_j^* \rangle = \delta_{ij}, \quad i, j = 1, 2, \dots, d,$$

where δ_{ij} again denotes the Kronecker delta. Similarly, given a set of linearly independent vectors

$$(6.27) \quad V = \{v_1, v_2, \dots, v_d\} \subset X,$$

a set of functionals

$$(6.28) \quad V^* = \{v_1^*, v_2^*, \dots, v_d^*\} \subset X^*$$

exists such that

$$(6.29) \quad \langle v_i^*, v_j \rangle = \delta_{ij}, \quad i, j = 1, 2, \dots, d,$$

whether X is reflexive or not.

Theorem 6.2. Suppose that $A \in L(X, Y)$ has a Fredholm theory, and

$$(6.30) \quad u^* A = Av = 0$$

if and only if $u^* \in \text{span}\{u_1^*, u_2^*, \dots, u_d^*\} \subset Y^*$ and $v \in \text{span}\{v_1, v_2, \dots, v_d\} \subset X$, where the defect d of A is positive. Then, for $u_k \in U$ and $v_k^* \in V^*$, $k = 1, 2, \dots, d$, where U and V^* are defined by (6.24)-(6.29), the operator

$$(6.31) \quad B = A - \sum_{k=1}^d u_k \langle v_k^*, \cdot \rangle$$

is invertible, and

$$(6.32) \quad AB^{-1}A = A,$$

so that $A^\dagger = B^{-1}$ is an inner inverse of A .

Proof: To show that B is invertible, consider the homogeneous equation $Bz = 0$, which is equivalent to

$$(6.33) \quad Az = \sum_{k=1}^d u_k \langle v_k^*, z \rangle.$$

As this equation is solvable if and only if the right-hand side is orthogonal to $u_1^*, u_2^*, \dots, u_d^*$ because A has a Fredholm theory, it follows from (6.26) that

$$(6.34) \quad \langle v_k^*, z \rangle = 0, \quad k = 1, 2, \dots, d,$$

and thus $Az = 0$. This means that z is of the form $z = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_d v_d$, where the coefficients α_k are given by (6.34), and hence $z = 0$ is the unique solution of the homogeneous equation $Bz = 0$, which implies the existence of B^{-1} .

To prove (6.32), note that from (6.29), (6.30), and (6.31),

$$(6.35) \quad Bv_i = - \sum_{k=1}^d u_k \langle v_k^*, v_i \rangle = -u_i, \quad i = 1, 2, \dots, d,$$

hence

$$(6.36) \quad B^{-1}u_k = -v_k, \quad k = 1, 2, \dots, d,$$

and

$$(6.37) \quad B^{-1}A = B^{-1}(B + \sum_{k=1}^d u_k \langle v_k^*, \cdot \rangle) = I - \sum_{k=1}^d v_k \langle v_k^*, \cdot \rangle,$$

and (6.32) follows directly from (6.30). QED

Instead of (6.36), one could also use the relationships

$$(6.38) \quad v_k^* B^{-1} = -u_k^*, \quad k = 1, 2, \dots, d,$$

to establish (6.32). The operator $B^{-1} = A^\dagger$ obtained from (6.31) is called *Hurwitz pseudoinverse* of A [31, pp. 165-168; 12], which goes back to 1912.

By the same reasoning as above, any operator of the form

$$(6.39) \quad A^\dagger = (A - \sum_{k=1}^d u_k \langle v_k^*, \cdot \rangle \beta_k \langle v_k^*, \cdot \rangle)^{-1}$$

for $\beta_1, \beta_2, \dots, \beta_d$ such that $\beta_1 \beta_2 \dots \beta_d \neq 0$ will be an inner inverse of A . However, as these operators are invertible, they cannot satisfy condition (2) of §1b which characterizes outer inverses; consequently, the construction (6.39), while useful for some purposes, only gives a partial solution to the problem of finding generalized inverses.

Another matter of computational importance relates to the calculation of generalized inverses of perturbations of operators with known generalized inverses. Suppose, for example, that one has an efficient technique to obtain the Moore-Penrose generalized inverse A^+ of A [27], and then would like to use the result to obtain the generalized inverses of perturbed operators $B = A + \Delta A$ with less effort than calculating B^+ *ab initio*, or error bounds for the approximation of B^+ by A^+ . As A^+ is not a continuous function of A in general, it would be expected that analytic perturbation methods apply only under restrictive conditions, as even for $\|\Delta A\|$ arbitrarily small, one of the algebraic catastrophes that the rank of B is greater or less than the rank of A could occur. Most applications of analytic perturbation theory to the above problems are carried out under assumptions that ensure $\text{rank}(B) = \text{rank}(A)$, or that the change in rank is known [23, pp. 333-351]. Algebraic perturbation methods, on the other hand, are not necessarily subject to this kind of limitation. For rank one modifications of A , C. D. Meyer, Jr. [19; 23, pp. 351-352] has obtained formulas of the type

$$(6.40) \quad (A + u > < v)^+ = A^+ + G$$

for all six possible cases, where G depends on A^+ and the data. More general finite-rank modifications (5.42) of A can then be handled by the method of successive rank one modifications corresponding to (5.51)-(5.52). This latter algorithm was originated by Greville [9] for the recursive calculation of the Moore-Penrose generalized inverse of a matrix. Formula (6.40) reduces to (5.28) in the special case that A is invertible, as for any generalized inverse of A , $A^+ = A^{-1}$ for all $A \in \mathcal{J}$. This suggests the computational strategy of using a method for generalized inversion on an operator which is suspected of being singular or nearly singular. If the operator or the perturbed operator actually involved in the calculation is nonsingular, then this technique will yield its inverse, whereas a straightforward inversion method might fail.

Another approach to ill-posed problems is to approximate them by a perturbed problem which is well conditioned. An example is the technique of *regularization*, due to A. N. Tihonov [39, 40], which has close connections with the subject of generalized inverses [22]. If the operator A in (1.1) does not have a bounded inverse, then the smallest perturbation Δy in the data can cause an enormous change Δx in

the solution of the perturbed problem (2.3) as compared to the solution of the reference problem. A typical situation in which problems of this type arise in applications is that X and Y are Hilbert spaces, and $A = K$ is a compact operator. The prototype of the resulting equation

$$(6.41) \quad Kx = y, \quad y \in Y,$$

is the linear Fredholm integral equation of the first kind,

$$(6.42) \quad \int_0^1 K(s,t)x(t)dt = y(s), \quad 0 \leq s \leq 1.$$

As perturbations in (6.42) in actual practice are inevitable, due to errors of measurement, discretization, and computation, direct numerical solution of (6.42) by standard techniques that work well for the integral equation (4.5) of second kind are rarely successful. The same observation may be made for (6.41) as compared to

$$(6.43) \quad (\alpha I - K)x = y$$

for $\alpha \neq 0$. In order to find an acceptable approximate solution of the perturbed version of (6.41), the method of regularization consists of finding an element $w(\alpha) \in X$ which minimizes the functional

$$(6.44) \quad f(w;\alpha) = \|Kw - z\|^2 + \alpha^2 \|w\|^2.$$

Thus, (6.44) represents a trade-off between the fidelity with which the perturbed equation $Kw = z$ is satisfied, and the size of the norm of the corresponding solution. The parameter α (or sometimes α^2) in (6.44) is called the regularization parameter. The crucial problem in this field is the determination of the optimal regularization parameter, for which the value of $f(w;\alpha)$ is minimum, or at least a method for obtaining good approximations to the optimal value. A significant recent advance in this area is the application by Grace Wahba [41] of the method of weighted cross-validation to the case that the perturbation is due to discretization of the data with random errors of the type known as "white noise".

7. The eigenvalue-eigenvector problem. As stated in §1c, this problem is to find eigenvalues λ and right eigenvectors $x \neq 0$ satisfying (1.9), where $A \in L(X,X)$, X a Hilbert space. It follows that one is interested in the values of λ for which the linear operator

$$(7.1) \quad T(\lambda) = A - \lambda I$$

is singular, and one may also want to find the left eigenvectors $y \neq 0$ of A corresponding to the eigenvalue λ which satisfy the homogeneous equation

$$(7.2) \quad y(A - \lambda I) = 0.$$

The additional assumption will be made that the values of λ considered are restricted to those for which $T(\lambda)$ has a Fredholm theory. This condition does not exclude any λ in the finite-dimensional algebraic case; however, for Fredholm integral operators of the first kind or compact operators in general, it is customary

to formulate the eigenvalue-eigenvector problem in terms of the reciprocal eigenvalues $\mu = 1/\lambda$, as the operator

$$(7.3) \quad S(\mu) = I - \mu K, \quad K \in \mathcal{K},$$

will have a Fredholm theory for all scalars $\mu \in \Lambda$ by Theorem 5.3. This is equivalent to excluding $\lambda = 0$ from consideration in (7.1) if A is compact.

In order to contemplate the application of analytic perturbation methods to the eigenvalue-eigenvector problem, it is essential to determine conditions under which this problem is well-posed, as the operator $T(\lambda)$ will be singular if λ is an eigenvalue. One way to do this is to convert equation (1.9) and the normalization condition (1.10) into the nonlinear system

$$(7.4) \quad P(q) := \begin{pmatrix} Ax - \lambda x \\ \frac{1}{2} - \frac{1}{2} \langle x, x \rangle \end{pmatrix} = 0$$

in the product space $Q = X \times \Lambda$ of vectors $q = (x, \lambda)^T$, $x \in X$, $\lambda \in \Lambda$. Suppose that $q_1 = (x_1, \lambda_1)^T$ is a solution of (7.4); that is, λ_1 is an eigenvalue of A , and x_1 is a corresponding normalized eigenvector. Then, the implicit function theorem [10] guarantees continuous dependence of the solution of (7.4) on the data if the linear operator $P'(q_1) \in L(Q, Q)$ has a bounded inverse, where $P'(q)$ is the Fréchet derivative

$$(7.5) \quad P'(q) = \begin{pmatrix} A - \lambda I & -x \\ -\langle x & 0 \end{pmatrix}$$

of the operator P at q [30, pp. 97-100]. The formulation (7.4), while not the most general [1], has the advantage that if A is Hermitian ($A^* = A$ [38, pp. 324-327]), then so is $P'(q)$. The following theorem gives an explicit formulation of the inverse operator $[P'(q_1)]^{-1}$ in this case if the defect of $T(\lambda)$ is equal to one, that is, if all solutions x of the homogeneous equation $T(\lambda_1)x = 0$ are scalar multiples of the normalized eigenvector x_1 , making use of the fact that the right and left eigenvectors of an Hermitian operator can be identified.

Theorem 7.1. If A is Hermitian, $q_1 = (x_1, \lambda_1)^T$ satisfies (7.4), and the defect of $T(\lambda_1)$ is equal to one, then

$$(7.6) \quad [P'(q_1)]^{-1} = \begin{pmatrix} B_1^{-1} - x_1 \langle x_1 & -x_1 \rangle \\ -\langle x_1 & 0 \end{pmatrix}$$

where

$$(7.7) \quad B_1^{-1} = (A - \lambda_1 I - x_1 \langle x_1)^{-1}$$

is the Hurwitz pseudoinverse of $A - \lambda_1 I$.

Proof: It follows by direct calculation and the use of (6.36) and (6.38) that

$$(7.8) \quad P'(q_1) \begin{pmatrix} B_1^{-1} - x_1 > < x_1 & -x_1 > \\ - < x_1 & 0 \end{pmatrix} = \begin{pmatrix} B_1^{-1} - x_1 > < x_1 & -x_1 > \\ - < x_1 & 0 \end{pmatrix} P'(q_1) = \\ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

the identity operator in $Q = X \times \Lambda$. QED

By the use of Theorem 6.2, formula (7.6) can be extended immediately to the non-Hermitian case $y_1 T(\lambda_1) = T(\lambda_1) x_1 = 0$, provided the defect of $T(\lambda_1)$ remains equal to one [1, §3]. Under these circumstances, results are available by the methods of analytic perturbation theory similar to those for nonsingular linear equations (1.1) [1, §5].

For the finite-dimensional case, perturbation methods and error analysis for the algebraic eigenvalue problem have been presented in great detail in the comprehensive work by J. H. Wilkinson [44, pp. 62-188]. Just one of these results will be cited here, which fits into the framework of algebraic perturbation theory. Suppose that w is a unit vector, and $p = (w, \mu)^T$ is an approximate solution of (7.4), so that

$$(7.9) \quad (A - \mu I)w = r,$$

with residual vector r . From equation (5.38), it follows that

$$(7.10) \quad (A - r > < w^* - \mu I)w = 0,$$

so that w is an exact eigenvector of the perturbed operator

$$(7.11) \quad B = A - r > < w^*$$

corresponding to the eigenvalue μ [44, pp. 170-171]. The perturbed operator B is simply a rank one modification of the reference operator A .

Another application of algebraic perturbation theory to the eigenvalue-eigenvector problem has been given by W. Stenger [37] to find inequalities between eigenvalues of perturbed and reference integral operators.

8. Linear programming. The solution of linear programming problems as formulated in §1d is one of the primary tools for decision making in government and commerce at the present time [8]. The number of variables involved is typically large, and a lot of computer time is expended for this purpose. Thus, an application of perturbation theory which would increase efficiency could result in substantial savings. Once again, the fact that the solutions do not depend continuously on the data in general limits the applicability of analytic perturbation techniques. A necessary and

sufficient condition for continuous dependence of the solution of the primal and dual linear programming problems in a neighborhood of solvable reference problems has been given recently by S. M. Robinson [34]. Studies of what is called *parametric programming* give conditions under which the solution of the reference problem remains unchanged under perturbation of the data [8, pp. 144-154]. On the subject of error estimation, P. Wolfe [45] has contributed a method for error analysis and control in the solution of linear programming problems.

Although changes in the objective function (1.14) are not usually difficult to deal with, perturbations in the constraints (1.15), as would result, for example, by the introduction of a new technology in an industry, may require the complete re-starting of the solution method used. Consequently, the following problem may be of practical interest.

Problem 8.1. Given the solution x of (1.14)-(1.15) and the associated information, such as the choice of pivots in the simplex algorithm [45], find an efficient method for solving

$$(8.1) \quad \text{minimize } f(w) := \langle d, w \rangle + \eta$$

subject to

$$(8.2) \quad Bw \leq z, \quad w \geq 0,$$

where all perturbations in the reference data are of finite rank which is small compared to the size of the reference problem.

9. Nonlinear problems. Although this survey has been concerned mainly with linear problems, it should be mentioned that perturbation methods are widely applied to the solution of nonlinear operator equations

$$(9.1) \quad P(x) = 0,$$

where P maps X into Y , and also *fixed point problems* in X of the form

$$(9.2) \quad x = H(x).$$

(It is evident that (9.2) is a special case of (9.1); conversely, there are many ways to convert (9.1) into an equivalent fixed point problem.)

These problems are well-posed in the neighborhood of a solution x_0 if, for example, H is continuous and contractive [30, Chapter 2], or, more restrictively, if P is differentiable and

$$(9.3) \quad \Gamma_0 = [P'(x_0)]^{-1} \in L(Y, X).$$

Depending on the smoothness of P , in this case one can base analytic perturbation techniques on the implicit function theorem [10], Newton's method and its variants, Taylor series expansions, inversion of power series, and so on [30, Chapter 4]. These methods are all essentially derived from the corresponding ideas of elementary scalar calculus.

Recently, W. Rheinboldt has given generalizations of the condition numbers (5.8) and (5.9) for nonlinear operators for which (9.3) holds, and a corresponding generalization of the perturbation formula (5.13) for error estimation [33].

Algebraic perturbation methods for nonlinear operator equations are less well investigated. A nonlinear operator F with range belonging to the finite-dimensional space

$$(9.4) \quad Y_n = \text{span} \{y_1, y_2, \dots, y_n\}$$

will be of the form

$$(9.5) \quad F(\cdot) = \sum_{j=1}^n y_j > f_j(\cdot),$$

where $f_1(\cdot), f_2(\cdot), \dots, f_n(\cdot)$ are (generally nonlinear) functionals on X . The perturbed operator equation

$$(9.6) \quad Q(x) = 0,$$

where $Q = P - F$, is equivalent to the equation

$$(9.7) \quad k(x) = \sum_{j=1}^n \xi_j y_j,$$

where

$$(9.8) \quad \xi_j = f_j(x), \quad j = 1, 2, \dots, n.$$

Suppose, and this is the *big assumption*, that the equation $P(x) = y$ is solvable for $y \in Y_n$, that is, an operator G is known which gives

$$(9.9) \quad x = G(\xi_1, \xi_2, \dots, \xi_n)$$

if $P(x) = y$ is of the form (9.7). Then, applying f_1, f_2, \dots, f_n in turn to (9.9) yields the nonlinear system

$$(9.10) \quad \xi_i = h_i(\xi_1, \xi_2, \dots, \xi_n), \quad i = 1, 2, \dots, n,$$

where $h_1 = f_1 G, h_2 = f_2 G, \dots, h_n = f_n G$, which is a finite-dimensional fixed-point problem in Λ^n of the form (9.2). On the basis of the *additional assumption* that

(9.10) is solvable, the substitution of its solutions $\xi_1, \xi_2, \dots, \xi_n$ into (9.9) provides a solution x of the nonlinear operator equation (9.6). As an example of this approach, the *Hammerstein integral equation* with kernel (5.31)

$$(9.11) \quad x(s) - \int_0^1 K(s, t) \phi(t, x(t)) dt = 0$$

is a rank one modification of the *nonlinear Volterra integral equation*

$$(9.12) \quad x(s) - \int_0^s L(s, t) \phi(t, x(t)) dt = 0$$

with kernel (5.33). Thus, if one can solve

$$(9.13) \quad x(s) - \int_0^s L(s, t) \phi(t, x(t)) dt = \xi u(s),$$

where

$$(9.14) \quad \xi = \int_0^1 v(t) \phi(t, x(t)) dt.$$

for $x(s) = g(s; \xi)$, then from (9.14), the system (9.10) is equivalent to the scalar fixed point problem

$$(9.15) \quad \xi = h(\xi) := \int_0^1 v(t) \phi(t, g(t; \xi)) dt ,$$

which is one nonlinear equation in one unknown [32, §5].

Although quite a bit is known about nonlinear systems (9.10) in finite-dimensional spaces [28], the theory and practice of their solution is far from the highly developed technology available for finite linear systems (5.44). There is also the ever-present big assumption. Even though (9.9) is not obtainable explicitly, the form of the problem (9.7) suggests iteration: Solve (9.7) for given $\xi_1^{(0)}, \xi_2^{(0)}, \dots, \xi_n^{(0)}$, substitute into (9.10) to obtain

$$(9.16) \quad \xi_i^{(1)} = h_i(\xi_1^{(0)}, \xi_2^{(0)}, \dots, \xi_n^{(0)}), \quad i = 1, 2, \dots, n ,$$

and so on. In the case that (9.6) is a boundary-value problem for a nonlinear differential equation, this is called "shooting" [13, Chapter 2, also §6.1]. Of course, this iteration may not converge, and some other method for solving (9.6) may be more appropriate.

This section will also conclude with an important problem, as much more work needs to be done.

Problem 9.1. For differentiable P , develop existence theory and find effective techniques for computing solutions x_0 of the nonlinear operator equation (9.1) in the case that $P'(x_0)$ does not have a bounded inverse.

REFERENCES

1. Anselone, P.M. and Rall, L. B., The solution of characteristic value-vector problems by Newton's method, Numer. Math. 11 (1968), 38-45. MR 36, #4795.
2. Banach, S., Théorie des Opérations Linéaires, Warsaw, 1932. Reprinted by Chelsea, New York, 1963.
3. Ben-Israel, A. and Greville, T. N. E., Generalized Inverses: Theory and Applications, John Wiley & Sons, New York, 1974. MR 53, #469.
4. Courant, R. and Hilbert, D., Methods of Mathematical Physics, Vol. 1., Interscience, New York, 1953.
5. Forsythe, G. E. and Moler, C. B., Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, N. J., 1967. MR 36, #2306.
6. Fredholm, I., Sur une classe d'équations fonctionnelles, Acta Math. 27 (1903), 365-390.
7. Friedman, Bernard, Principles and Techniques of Applied Mathematics, John Wiley & Sons, New York, 1956. MR 18, 43.
8. Gass, S. J., Linear Programming. Methods and Applications, 3rd Ed., McGraw-Hill, New York, 1969. MR 42, #1509.
9. Greville, T. N. E., Some applications of the pseudoinverse of a matrix, SIAM Rev. 2 (1960), 15-22. MR 22, #1067.

11. Householder, A. S., The Theory of Matrices in Numerical Analysis, Blaisdell, New York, 1964. MR 30, #5475.
12. Hurwitz, W. A., On the pseudo-resolvent to the kernel of an integral equation, Trans. Amer. Math. Soc. 13 (1912), 405-418.
13. Keller, H. B., Numerical Methods for Two-Point Boundary-Value Problems, Blaisdell, Waltham, Massachusetts, 1968. MR 37, #6038.
14. Lonseth, A. T., Systems of linear equations with coefficients subject to error, Ann. Math. Statistics 13 (1942), 332-337. MR 4, 90.
15. Lonseth, A. T., On relative errors in systems of linear equations, Ann. Math. Statistics 15 (1944), 323-325. MR 6, 51.
16. Lonseth, A. T., The propagation of error in linear problems, Trans. Amer. Math. Soc. 62 (1947), 193-313. MR 9, 192.
17. Lonseth, A. T., An extension of an algorithm of Hotelling, Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability, 1945, 1946, pp. 353-357. University of California Press, Berkeley and Los Angeles, 1949. MR 10, 627.
18. Lonseth, A. T., Sources and applications of integral equations, SIAM Rev. 19 (1977), 241-278.
19. Meyer, C. D., Jr., Generalized inversion of modified matrices, SIAM J. Appl. Math. 24 (1973), 315-323. MR 47, #5010.
20. Nashed, M. Z., Generalized inverses, normal solvability, and iteration for singular operator equations, Nonlinear Functional Analysis and Applications, ed. by L. B. Rall, pp. 311-359, Academic Press, New York, 1971. MR 43, #1003.
21. Nashed, M. Z. (Editor), Generalized Inverses and Applications, Academic Press, New York, 1976.
22. Nashed, M. Z., Aspects of generalized inverses in analysis and regularization, Generalized Inverses and Applications, ed. by M. Z. Nashed, pp. 193-244, Academic Press, New York, 1976.
23. Nashed, M. Z., Perturbations and approximations for generalized inverses and linear operator equations, Generalized Inverses and Applications, ed. by M. Z. Nashed, pp. 325-396, Academic Press, New York, 1976.
24. Nashed, M. Z. and Rall, L. B., Annotated bibliography on generalized inverses and applications, Generalized Inverses and Applications, ed. by M. Z. Nashed, pp. 771-1041, Academic Press, New York, 1976.
25. Nashed, M. Z. and Votruba, G. F., A unified operator theory of generalized inverses, Generalized Inverses and Applications, ed. by M. Z. Nashed, pp. 1-109, Academic Press, New York, 1976.
26. Noble, B., Applied Linear Algebra, Prentice-Hall, Englewood Cliffs, N.J., 1969. MR 40, #153.
27. Noble, B., Methods for computing the Moore-Penrose generalized inverse, and related matters, Generalized Inverses and Applications, ed. by M. Z. Nashed, pp. 245-301, Academic Press, New York, 1976.
28. Ortega, J. M. and Rheinboldt, W. C., Iterative Solution of Non-linear Equations in Several Variables, Academic Press, New York, 1970. MR 42, #8686.

29. Rall, L. B., Error bounds for iterative solution of Fredholm integral equations, Pacific J. Math. 5 (1955), 977-986. MR 18, 72.
30. Rall, L. B., Computational Solution of Nonlinear Operator Equations, John Wiley & Sons, New York, 1969. MR 39, #2289.
31. Rall, L. B., The Fredholm pseudoinverse--an analytic episode in the history of generalized inverses, Generalized Inverses and Applications, ed. by M. Z. Nashed, pp. 149-173, Academic Press, New York, 1976.
32. Rall, L. B., Resolvent kernels of Green's function kernels and other finite rank modifications of Fredholm and Volterra kernels, J. Optimization Theory Appl. (to appear). Preprint: MRC TSR#1670, University of Wisconsin-Madison, 1976.
33. Rheinboldt, W. C., On measures of ill-conditioning for nonlinear equations, Math. Comp. 30 (1976), 104-111.
34. Robinson, S. M., A characterization of stability in linear programming, Operations Res. (to appear). Preprint: MRC TSR#1542, University of Wisconsin-Madison, 1975.
35. Sherman, J. and Morrison, W. J., Adjustment of an inverse matrix corresponding to a change in one element of a given matrix, Ann. Math. Statistics 21 (1950), 124-127. MR 11, 693.
36. Shieh, A. S., On the numerical solution of Poisson's equation by a capacitance matrix method, Proceedings of the 1977 Army Numerical Analysis and Computers Conference, U. S. Army Research Office, Research Triangle Park, N.C. (to appear).
37. Stenger, W., On perturbations of finite rank, J. Math. Anal. Appl. 28 (1969), 625-635.
38. Taylor, A. E., Introduction to Functional Analysis, John Wiley & Sons, New York, 1958. MR 20, #5411.
39. Tihonov, A. N., On the solution of ill-posed problems and the method of regularization (Russian), Dokl. Akad. Nauk SSSR 151 (1963), 501-504. MR 28, #5576.
40. Tihonov, A. N., On the regularization of ill-posed problems (Russian), Dokl. Akad. Nauk SSSR 153 (1963), 49-52. MR 28, #5577.
41. Wahba, Grace, Practical approximate solutions to linear operator equations when the data are noisy, SIAM J. Numer. Anal. 14 (1977) (to appear). Preprint: Department of Statistics Technical Report #430, University of Wisconsin-Madison, 1975.
42. Widlund, O. and Proskurowski, W., On the numerical solution of Helmholtz's equation by the capacitance matrix method, ERDA Rep. C00-3077-99, Courant Institute of Mathematical Sciences, New York University, New York, 1975.
43. Wilkinson, J. H., Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, N. J., 1963. MR 28, #4661.
44. Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965. MR 32, #1894.
45. Wolfe, P., Error in the solution of linear programming problems, Error in Digital Computation, Vol. II, ed. by L. B. Rall, pp. 271-284, John Wiley & Sons, New York, 1965. MR 32, #4830.
46. Woodbury, M. A., Inverting modified matrices, Statistical Research Group, Memo. Rep. no. 42, Princeton University, Princeton, N. J., 1950. MR 12, 361.

SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS
ON VECTOR COMPUTERS

James M. Ortega
Robert G. Voigt

ABSTRACT

In this paper we review the present status of numerical methods for partial differential equations on vector computers, that is, computers with hardware instructions for vector operands. Both direct and iterative methods are for elliptic equations as well as explicit and implicit methods for initial boundary value problems. The intent is to point out attractive methods, as well as areas where this class of computer cannot be fully utilized because of either hardware restrictions or the lack of adequate algorithms.

This paper is the text of an invited survey talk presented at the 1977 Army Numerical Analysis and Computers Conference. The preparation of this paper was supported under NASA Contract NAS1-14101 while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, VA. 23665

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

6 OF 7
ADA
046552



I. Introduction

For the past 15 years, there has been interest in the use of computers with a parallel or pipeline architecture for the solution of very large scientific computing problems. As a result of the impending implementation of such computers, there was considerable activity in the mid and late 1960's in the development of parallel numerical methods. Some of this work is summarized in the classical review article of Miranker [1971]. It has only been in the period since then, however, that such machines have become available. The ILLIAC IV was put into limited operation at NASA's Ames Research Center in 1972; the first Texas Instrument Advanced Scientific Computer (TI-ASC) became operational in Europe in 1972 primarily for seismic calculations; the first Control Data Corporation STAR-100 was delivered to Lawrence Livermore Laboratory in 1974; and the first Cray Research Corporation CRAY-1 was put into service at Los Alamos Scientific Laboratory in 1976. (A summary of the basic characteristics of these four machines will be given in Section 2). There have also been a number of other parallel configurations designed primarily for other types of applications - such as Goodyear Corporation's STARAN (Goodyear [1974], Gilmore [1971], Rudolph [1972]) and the PEPE system (Berg, et al. [1972]) - or for research purposes, such as the C.mmp system at Carnegie-Mellon University (Wulf and Bell [1972]).

The ILLIAC IV, TI-ASC, CDC STAR-100, and CRAY-1, although differing sometimes considerably in their architectures, are all examples of vector computers; that is, they have hardware instructions which accept vectors as operands. With the exception of the ILLIAC IV, they also have the usual scalar operations. Under optimum conditions, these machines are capable of producing floating point results at rates up to 50 to 200 million per second. Here, "optimum conditions" vary somewhat amongst the machines but, roughly, it means

operating on vectors of fairly long lengths (10000 or longer) on the ASC or STAR-100, or of lengths which are multiples of 64 on the ILLIAC IV or CRAY-1. This will be elaborated on in the sequel.

As an example of the gain that can be achieved, experiments at Langley Research Center showed a speed-up of approximately 30 to 1 for the STAR-100 over a CDC 6600 on the solution of 300×300 linear systems by Gaussian elimination. Both codes were written in assembly language and made as optimal as possible.

In practice, it is difficult to achieve optimum conditions and the challenge for the numerical analyst is to devise algorithms which utilize as much as possible the fast vector processing capabilities. Stone [1973b] has given the following general dictums:

1. Data must be arranged (and possibly rearranged) for efficient computation.
2. Efficient serial algorithms are not necessarily good for parallel machines and, conversely, inefficient serial algorithms may lead to efficient parallel algorithms.
3. Some algorithms may appear to be inherently serial but may be transformed to efficient parallel algorithms.

There are as yet very few papers in print which describe in detail the application of vector computers to realistic problems and some of these appeared before the machines themselves were available. For example, Carroll and Weatherald [1967] discuss the possible application of the Solomon computer - the predecessor, which was never built, of the ILLIAC IV - to hydrodynamics problems and general circulation weather models in particular; Reilly [1970] considers a Monte Carlo method for the Boltzmann equation on the ILLIAC IV;

and Ogura, Sher, and Ericksen [1972] review the theoretical efficiency of the ILLIAC IV for hydrodynamics calculations.

More recently, Wilhelmson [1974], and Erickson and Wilhelmson [1976] have considered convection problems - and in particular the Benard-Rayleigh problem - on the ILLIAC IV; they use Dufort-Frankel differencing on the diffusion terms, a scheme of Lilly for the convection terms, a fast Fourier method for the Poisson equation, and leap-frog differencing in time. One of the main thrusts of their work is a proper balancing of computation with disk to core transfers.

Davy and Reinhart [1975] discuss the application of the ILLIAC IV to a chemically reacting, inviscid hypersonic flow problem, using MacCormack's method with shock capturing. McCulley and Zaher [1974] report on the solution of diffusion type equations on the ILLIAC IV in a problem that arises in planetary entry. Boris [1976] applies his flux-corrected transport (FCT) algorithm to continuity type equations on the TI-ASC; he concludes that the FCT method is "fully vectorizable."

Lambiotte and Howser [1974] compared the ADI method, Brailovskaya's method [1965], and Graves' Partial Implicitization method [1975] on the CDC STAR-100 for the driven cavity problem and conclude that both Brailovskaya's method and Graves' method vectorize well and are the fastest on the STAR-100 even though the ADI method was the fastest on a serial machine. Weilmunster and Howser [1976] consider a boundary layer/shock interaction calculation governed by the full Navier-Stokes equations in 2 dimensions. They report speed-ups of as much as 65 to 1 on the STAR over a corresponding program on a 6600 (but the 6600 program used the RUN compiler which produces object code that normally runs about twice as slow as the FTN compiled code and the speed-up is more correctly on the order of 30 or 35-1).

Much more work is now in progress at the various laboratories which have vector machines and we can hope to see many additional insights and comparisons in the near future. Rather than report in detail on any particular problems, it is the purpose of this paper to review some of the basic ideas involved in solving partial differential equations on vector computers. A related survey is by Heller [1977] which concentrates on linear algebra computations and gives more emphasis to theoretical questions about parallel computational processes. We mention also the annotated bibliography of Poole and Voigt [1974] which was an essentially complete listing of works pertaining to numerical methods for vector and parallel computers up to the time of its publication.

After summarizing the basic architectural features of vector computers in Section 2, we consider in Section 3 direct methods for the solution of the discrete systems of algebraic equations which result from elliptic boundary value problems. In Section 4, we treat iterative methods for elliptic equations as well as marching methods for initial-boundary value problems for parabolic and hyperbolic equations. Finally, in Section 5 we summarize very briefly our conclusions.

Throughout, we have tried to take as broad a view as possible, but since our own experience is primarily with the CDC STAR-100, many examples and details are necessarily given in terms of this machine.

2. Summary of Vector Hardware

For the purposes of this paper, we will consider a vector computer as one capable of operating on the contents of a collection of memory locations, known as a vector, with a single hardware instruction. The architecture of a specific computer dictates the amount of generality permitted in defining the elements of a vector. For example, as we will see below, some computers

allow vector operations on memory locations which are a constant multiple apart while others permit them only on contiguous memory locations. We will discuss in this section the basic characteristics of the ILLIAC IV, the CDC STAR-100, the TI ASC, and the CRAY-1, all of which have hardware vector operations. The latter three computers are discussed in detail in Ramamoorthy and Li [1977]. We have specifically excluded from consideration computers like the CDC 7600 which execute vector operations by means of software aids such as STACKLIB (see e.g. McMahon, Sloan and Long [1972]).

ILLIAC IV. The present manifestation of the original design for the ILLIAC IV (Barnes et al. [1968]) has 64 processing elements (PE's) each capable of executing the same instruction at the same time (Bouknight et al. [1972]). Each PE has 2048 64 bit words of local memory. The main memory of the ILLIAC consists of a fixed head disk with a capacity of about 16 million words. The disk has a rotation period of 40 milliseconds and is capable of transfer rates of about 960 million bits per second.* Consequently, the overriding consideration in using the ILLIAC IV is to formulate the problem and choose algorithms which permit predictable transfers of large blocks of data to and from the disk. The data transfer problem is eased by a PE interconnection pattern of an 8 x 8 grid that permits the movement of data in a PE to any one of its immediate north, south, east or west neighbors. In addition each PE on the boundary of the grid is connected to the corresponding PE on the opposite boundary (see Figure 1).

*Secondary storage on the STAR-100, ASC and CRAY-1 consists of multiple disk storage units with similar characteristics. In all cases the disk capacity is on the order of 10^9 bits, the latency is approximately 15 msec, and the transfer rate is in excess of 30×10^6 bits per second.

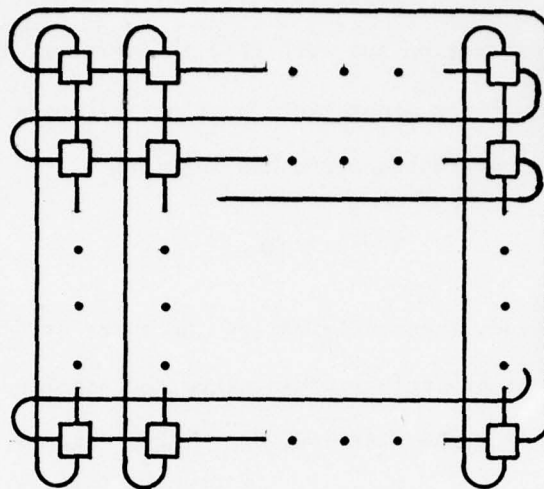


Figure 1. PE interconnection pattern for the ILLIAC IV

For the ILLIAC IV a vector consists of up to 64 elements with each element in the memory of a different PE. Operating on vectors of this type, the computer is capable of over 40 million floating point operations per second (MFLOPS). Using 32-bit mode this rate nearly doubles.

STAR-100. The CPU of the STAR-100 consists of two pipelines which are configured via microcode to execute the appropriate instruction (Control Data Corp. [1975]). A vector instruction initiates the flow of operands from memory to the pipeline. Assuming that the instruction involves two source vectors, each segment of the pipeline accepts two elements, performs its particular function (e.g. coefficient alignment or exponent adjustment), passes the result to the next segment, and receives the next two elements from the stream of operands. When the result of the operation emerges from the pipeline, it is returned directly to memory. For more details on pipeline

architecture see, for example, Stone [1975]. There is some overhead in initiating a vector instruction but once this is overcome, the pipeline produces a result about every minor cycle or clock. Thus a timing formula in minor cycles for vector instructions has the form

$$(2.1) \quad T = S + \alpha n$$

where S is the overhead, frequently called the start-up time, $1/\alpha$ is the number of results per minor cycle emerging from the pipeline, and n is the length of the vector. For the STAR-100, $S = 0(100)$ and $\alpha = 0(1)$. We note that n should actually be replaced in (2.1) by $8\lceil \frac{n}{8} \rceil$ which implies that vectors whose length is a multiple of 8 are the most efficient to work with. But, for simplicity, we will assume the timing formula is as given by (2.1).

To see more graphically the effect of start-up time, we give in Figure 2 a plot of the results per microsecond for 64-bit addition on the STAR-100 as a function of the vector length n . Here, $S = 71$, $\alpha = \frac{1}{2}$, and the minor cycle time is .04 μsec so that, from (2.1), the number of results per microsecond is given by $50n/(142+n)$, with the asymptotic limit equal to 50 MFLOPS

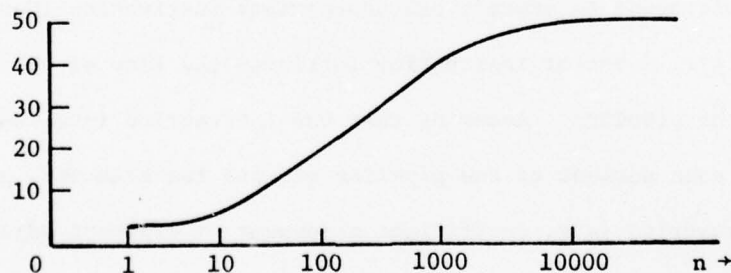


Figure 2.- Results per microsecond for the addition of two 64 bit vectors of varying length on the STAR-100.

From Figure 2, it is evident that even for vectors of length 100, the result rate is only 40% of that possible and that vectors of length almost 10,000 are needed before the start-up overhead becomes completely negligible. Note also that there is no sharp discontinuity in result rates as on ILLIAC IV where, for example, operations on vectors of length 65 would take twice as long as for vectors of length 64, since there are only 64 PE's.

The memory of the STAR-100 consists of up to one million 64 bit words. A vector is limited to contiguous memory locations. This restriction is eased somewhat by the rich instruction set which contains many operations for the manipulation of memory; however, as will be seen later, the cost of these operations can become a significant percentage of the execution time for a given algorithm.

An interesting feature of the STAR-100 is the ability to do floating point operations on halfwords of 32 bits. This has the effect of doubling the memory size and more than doubling the result rate. The STAR-100 can achieve over 100 MFLOPS in 32 bit mode.

We note that one shortcoming of the current STAR hardware - which will be considerably improved in the forthcoming 100A - is the rather slow time for scalar operations. The precise timings are difficult to ascertain since much depends on register conflicts, etc. but an average floating point operation time seems to be approximately 1 μ sec, which is roughly the same as the CDC 6600.

ASC. The CPU of the ASC consists of one, two or four pipelines which, as in the STAR-100, are configured for the desired operation (Texas Instruments [1974]). A vector instruction causes the operands to move from memory, through the pipelines and back to memory. The timing of vector instructions is given by equation (2.1), and as with the STAR-100, $S = O(100)$ and

$\alpha = O(1)$ although for one or two pipeline versions S may be in the range of 25 to 50 (see Texas Instruments [1974]).

A vector on the ASC is the contents of a collection of memory locations which can be referenced by a triply nested FORTRAN DO loop where the inner most loop has an increment of one; however, there may be a degradation in speed if the locations are not contiguous. This makes possible an interesting feature of the computer: it can translate an operation in DO loops nested up to three deep into a single hardware instruction. For example

```
DO 1 I = 2, 11
    DO 2 J = 1, 20, 2
        DO 3 K = 10, 1, -1
            A(I,J,K) = B(I,J,K) + C(I,J,K)
3        CONTINUE
2    CONTINUE
1 CONTINUE
```

becomes one vector add instruction for vectors of length 1000.

The importance of this is that there is only one start-up required in such a situation. As we will show later, this can have a pronounced effect on the performance of certain algorithms.

Memory on the ASC is available with up to 16 million 32 bit words. A four pipeline version of the computer is capable of performance in the range of 50 MFLOPS. As with the STAR-100 scalar arithmetic is relatively slow. The average time for a 32 bit floating point operation on a one pipeline computer appears to be about $.5 \mu$ sec. This can decrease by as much as a factor of four with the addition of more pipelines and assuming an instruction mix that can be executed in parallel.

CRAY-1. The CPU of the CRAY-1 contains twelve functional units which are segmented like the pipelines of the ASC and STAR-100; (Cray Research, Inc. [1976]). Unlike these pipelines, however, the functional units are not reconfigurable; each functional unit executes a subset of the instruction set. For performing vector operations the CRAY-1 has eight 64 word vector registers. A vector is defined as the contents of consecutive elements of one of the vector registers, always beginning with the first element of that register. The vector registers are loaded from memory, which consists of up to one million 64 bit words, and their contents returned to memory by vector load and store operations. Thus, as on the ILLIAC IV, a vector on the CRAY-1 is less than or equal to 64 words with operations on longer strings being done in increments of 64. Unlike the ILLIAC IV there may be little penalty for processing a vector of length 65 versus one of length 64. If a register is available, the element is loaded and the operation begins in the next cycle after the operation on the 64th element began. There may, however, be conflicts which degrade the performance from this ideal situation.

Two interesting features of the functional units are that they may operate independently and in parallel, and that they may be "chained." The latter feature means that the results of one functional unit may be used as input to another immediately without returning to the vector registers. These are particularly useful and powerful capabilities making it possible for the CRAY-1 to achieve in excess of 160 MFLOPS.

An outstanding feature of the CRAY-1 relative to the other vector computers is its fast scalar arithmetic. With a cycle time of 12.5 ns, the CRAY-1 is more than twice as fast on scalar operations as the CDC 7600 (see Keller [1976]).

3. Direct Methods for Elliptic Boundary Value Problems

Most of the work that has been done to date on the solution of elliptic boundary value problems on vector computers has centered around the solution of the algebraic system that results from discretizing the partial differential equation. Our discussion will reflect this emphasis, and include both direct and iterative methods for solving the algebraic systems.

If the discretization is by finite difference methods, then there is little computational work involved in generating the algebraic system; however, if finite element methods are used, there may be considerable effort required, and it is not clear that this computation is well suited for vector computers. Essentially the only paper that has addressed this point is Noor and Fulton [1975] who consider both the generation as well as the solution on the STAR-100 of a finite element structural analysis system. A procedure for generating the element stiffness matrix that utilizes vector operations is given; a timing estimate of this process indicates a speedup of at least a factor of 5 over the CDC-6600. The algorithm for factoring the linear system is similar to the banded ones described below.

In this section we will first consider algorithms for factoring the coefficient matrix, assuming that some natural ordering of the grid points is used. Next, we treat different orderings, especially those of one-way and nested dissection. Finally, we consider special methods for tridiagonal systems, and fast Poisson solvers based on the fast Fourier transform.

Factorization Methods. We make no specific assumptions about the differential equation, the domain or the discretization but assume only that the discrete equations

$$(3.1) \quad Ax = b$$

are such that the matrix A is symmetric, positive definite and banded, with bandwidth $\beta(A)$ defined by

$$\beta(A) \equiv \max_{a_{i,j} \neq 0} |i-j|.$$

We note that although A is banded it would be inefficient in many situations not to take advantage of its profile structure (see e.g. Jennings [1966]) but we shall not discuss this here. We consider the Cholesky decomposition $A = LL^T$ where L is lower triangular and no attempt is made to exploit the zeros within the band since the band itself almost completely fills during the factorization. The solution of (3.1) is then obtained by solving the systems $Ly = b$ and $L^Tx = y$. Algorithms are discussed for three computers — the STAR-100, the ASC, and the CRAY-1.

In Lambiotte [1975] several algorithms for performing the factorization of A were analyzed for a virtual computer patterned after the STAR-100. The following algorithm is a variant of one which was shown to be among the fastest. The lower half of the matrix A is assumed to be stored by columns, and the factor L and the modifications to A are developed a column at a time using the appropriate linear combinations of the columns of A . For example, the following vector operations are involved in the modification of column $i+j$ of A required when computing column i of L :

$$(3.2) \quad (a_{i+j,i+j}, \dots, a_{i+\beta,i+j}) \leftarrow (a_{i+j,i+j}, \dots, a_{i+\beta,i+j}) - a_{i+j,i} (a_{i+j,i}, \dots, a_{i+\beta,i}).$$

(The algorithm is not implemented in precisely the form given in (3.2); for details the reader is referred to Knight, Poole, and Voigt [1975] and George, Poole, and Voigt [1976b].) Expression (3.2) must be executed for j ranging from 1 to β ; hence, the vector lengths for the algorithm vary from 1 to β with an average of $\frac{\beta+1}{2}$. In George, Poole, and Voigt [1976b] a precise timing formula is given which, omitting scalar arithmetic times, is approximately

$$(3.3) \quad T(N, \beta) \approx \frac{3}{4} N \beta^2 + 232 N \beta + \text{lower order terms}$$

STAR-100 cycles. Note that the large coefficient of the $N\beta$ term causes that term to dominate the timing for all but very large problems. Most of the cycles of that coefficient are attributable to vector start-up times; as will be seen, this is in sharp contrast with the ASC where, by choosing a different algorithm, we will be able to reduce the influence of the start-up times to lower order terms. Note that even for large problems, if the bandwidth is small the algorithm will be very inefficient for it requires a large number of vector operations of short lengths. This effect reaches its extreme in solving tridiagonal systems, a topic we will discuss later.

In Calahan, Joy, and Orbits [1976] it was noted that the correct implementation on the ASC of an LU factorization of a full $n \times n$ matrix yields a timing formula where the start-up times contribute to the $O(n)$ term rather than the n^2 term as in the STAR-100. In Voigt [1977] that same improvement is obtained in the banded case by using an algorithm whose timing has the start-up time contributing to the N term rather than the $N\beta$ term as with the STAR-100. In essence we must store an additional band of zeros, nearly doubling the storage required by the above algorithm, and the operations have constant length β^2 , some of which are with zeros, rather than average length $\frac{\beta+1}{2}$. Assuming row by row storage, the key operations at the k^{th} step of the factorization are given by:


```

DO 10 J = K + 1, K +  $\beta$ 
      DO 20 I = K -  $\beta$ , K - 1
            TEMP = TEMP + A(K,I) * A(J,I)
      20 CONTINUE
10 CONTINUE

```

This nested DO loop can be executed on the ASC in one inner product instruction on vectors of length β^2 rather than 2β instructions of average length $\frac{\beta+1}{2}$ on the STAR-100. The timing formula for the factorization is then approximately

$$(3.4) \quad T(N, \beta) \approx N\beta^2 + \frac{19}{16}N\beta + 485N + \text{lower order terms}$$

ASC cycles. This formula assumes no degradation in performance for non-contiguous vector elements.

The usefulness of this algorithm clearly depends on N and β since we have doubled the coefficient of the $N^2\beta$ term in order to remove the start-up time influence to the $O(N)$ term. This tradeoff is discussed in more detail in Voigt [1977]. That paper also includes a discussion of the influence of individual operation times on the two machines, including the impact of a slow inner product on the STAR-100 versus a fast one for the ASC.

We now turn our attention to the CRAY where a different architectural characteristic is dominant. As was mentioned in Section 2, all vector instructions obtain their operands from the vector registers; thus these registers must be loaded from memory and this can have a major impact on the effective computational rate as has been noted by Calahan, Joy and Orbits [1976] and Fong and Jordan [1976]. To be specific, there is one path between memory and the vector registers capable of transferring one operand or result per clock. With the functional units capable of performing an add and multiply every

clock, the potential problem is clear. For example, if the STAR-100 algorithm given in expression (3.2) were implemented, it is shown in Voigt [1977] that the arithmetic units would be busy only half of the time because a load and a store are required for each multiply-subtract combination.

By redesigning the algorithm it is possible to keep both arithmetic units busy. Following the spirit of algorithms presented in Calahan, Joy and Orbits [1976] and Fong and Jordan [1976], this is accomplished by completely modifying the $k+1^{\text{st}}$ column of the matrix at the k^{th} step of the factorization and leaving all other columns unchanged. The key portion of the algorithm is given below where $j = 1, \dots, \beta$:

$$(3.5) \quad (a_{k+1,k+1}, \dots, a_{k+j,k+1}) \leftarrow (a_{k+1,k+1}, \dots, a_{k+j,k+1}) \\ - a_{k-\beta+j,k+1} * (a_{k+1,k-\beta+j}, \dots, a_{k+j,k-\beta+j})$$

Further details may be found in Voigt [1977] but note that the $k+1^{\text{st}}$ column need not be stored until the loop on j is completed. Thus the arithmetic units can be kept busy essentially throughout the factorization.

In Calahan, Joy and Orbits [1976] algorithms similar to those given in expressions (3.2) and (3.5) for factoring a dense matrix are compared with runs on a CRAY-1 computer and the times reported there bear out the contention that algorithm (3.5) is twice as fast as algorithm (3.2). In Orbits and Calahan [1976] a block method is discussed for factoring a dense system which requires about $\frac{1}{5}$ of the memory-register transfers that are required of the best algorithm discussed above. The algorithm is very complicated to implement on the CRAY-1 and will not be discussed here since we have already considered one which is limited by arithmetic unit speed rather than memory-register transfer speed.

This block algorithm, as well as its usefulness in the banded case, also is discussed in Fong and Jordan [1976].

The previous discussion applies to arbitrary banded systems of equations and, in particular, to the discrete forms of elliptic equations in which the ordering of the node points is such as to give rise to a suitably banded system. We now consider other node ordering schemes that have been shown to reduce the number of arithmetic operations required for factoring the matrix of the resulting linear system. The first of these is known as one-way dissection and is discussed in detail in George [1972, 1977]. Referring to Figure 3, the idea is first to divide the grid with ℓ horizontal separators. The nodes in the $\ell+1$ remaining rectangles are numbered vertically toward a separator and then the separators are numbered.

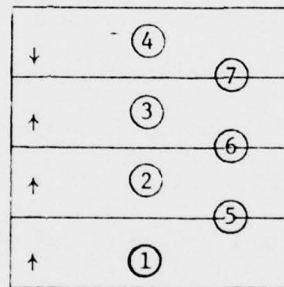


Figure 3. An n by n mesh dissectioned into 4 blocks with the ordering indicated by the circled numbers and the arrows.

For the proper choice of ℓ this ordering has been shown to reduce the number of arithmetic operations required for the factorization from $O(n^4)$ for the natural ordering to $O(n^{\frac{7}{2}})$ (see George [1972]).

The nested dissection ordering further reduces the operation count to $O(n^3)$ as shown in George [1973, 1977]. The idea here is to divide the grid with both horizontal and vertical separators as shown in Figure 4.

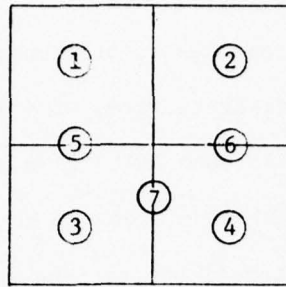


Figure 4. One step of the nested dissection ordering for the n by n grid.

Now regions 1 - 4 may be numbered in the usual way or, since they are again squares, they may be numbered using horizontal and vertical separators. Clearly the idea may be applied recursively, and in the case $n = 2^k - 1$, dissection will terminate after $k-1$ steps. In order to obtain the $O(n^3)$ operation count, dissection must be carried to completion; however as noted in George, Poole, and Voigt [1976a], there are advantages to stopping the dissection early.

Nested dissection for vector computers was first discussed by Calahan [1975], who considered rather general rectangular finite elements. The paper includes estimates of the number of vector operations required for the factorization and their average lengths assuming dissection is carried to completion. Based on this information it is concluded that nested dissection would be attractive on a vector computer such as the STAR-100 or the ASC. The idea of stopping the dissection early is not considered.

The appropriate level of dissection becomes an interesting question for a vector computer. We have already seen that, because of start-up times, it is desirable to work with vectors whose length is as great as possible;

however, it should be clear from Figures 3 and 4 that as dissection continues at least some of the vectors get shorter. This phenomenon is studied in detail in George, Poole, and Voigt [1976b], and there are two results worth mentioning. First, as one would expect, efficient use of vector computers dictates less dissection than implied by the serial operation count. For example, on computers for which the ratio of start-up time to result time is similar to that of the STAR-100 or the ASC, the minimum time for a given factorization is obtained by stopping nested dissection two levels from completion.

Secondly, both the one-way and nested dissection algorithms translate almost entirely into vector operations; however, in spite of a lower operation count, one-way dissection actually introduces more vector operations than are present in the natural ordering and this results in the natural ordering being superior for all but very large n . For example, on the STAR-100 the crossover point does not occur until n is approximately 650 whereas in the scalar case one-way dissection is superior for $n \geq 30$. In contrast, nested dissection not only reduces the operation count but it can also be implemented with fewer vector operations than the natural ordering. Further details on this comparison are also given in Voigt [1977].

In principle, both dissection algorithms would be attractive on the CRAY-1 and the ILLIAC IV since neither computer is burdened with vector start ups. However, both computers are limited by their I/O capabilities and we have already seen how this can limit the effectiveness of the CRAY-1. As was shown in George [1972, 1973], one-way and nested dissection reduce the storage requirements from $O(n^3)$ locations for the natural ordering to $O(n^{\frac{5}{2}})$ and $O(n^2 \log n)$ respectively. Thus it would appear that there would be less I/O demanded by the orderings. However, to date, there has

been no careful analysis of the I/O required by an implementation of either of the orderings on a computer such as the CRAY-1 or the ILLIAC IV.

One difficulty with the dissection orderings that should not be minimized is that they are at best difficult to apply to nonrectangular domains. Although it is usually clear how to proceed, the implementation of an automatic procedure for performing the ordering is a difficult problem in computer software. A step in this direction has been taken by George and Liu [1972] for conventional serial computers, but the techniques they employ do not seem to be very well suited for vector computers. In principle an irregular domain causes no great difficulties in implementing the natural ordering; however, because of the possible irregular band structure, it may be desirable to use the so-called profile method (see, e.g. Jennings [1966]) which takes advantage of the leading zeros in each individual row (or column).

Another factorization technique, called the hypermatrix scheme, was developed by von Fuchs, et al. [1972], who were motivated by the finite element analysis of large, complex structures which produce global structural matrices that do not fit in the main storage of a computer. The actual factorization is simply block decomposition; the unique feature of the scheme is that a hierarchy of bit matrices is used to manage the nonzero entries of the matrix. At the first level a bit matrix is used to identify the blocks which have nonzero entries and thus must be used in the computation. For large problems it may be inconvenient or impossible to keep even the bit matrix in memory; hence, a second bit matrix is introduced whose entries indicate blocks of the first bit matrix that have any "ones" present. This may be continued to any desired depth. In Noor and Voigt [1975] this approach is analyzed for the STAR-100 and a vectorized block

factorization algorithm is given. It is shown that the ability to select the block size can be beneficial not only to the structural analyst, but also in maximizing the efficiency of the computer. For example, the block size could be chosen so that the time required for the computations on the blocks overlapped the time required to bring a new block in from, and write an old block out to, secondary storage. This balance of I/O and computation is known to be a significant problem for vector computers (see, for example, Knight, Poole, and Voigt [1975] and Orbits and Calahan [1976]).

We have mentioned above that the average vector lengths occurring in the algorithms for factorization of a matrix with bandwidth β are $O(\beta)$. The tacit assumption has been that β is sufficiently large so that these operations are reasonably efficient; however, that is not always the case. For example, in iterative methods such as ADI or SLOR one must solve a large number of tridiagonal systems, that is, systems with $\beta = 1$. Since the previous algorithms are totally inappropriate for this problem we will now discuss some different algorithms for the factorization of tridiagonal matrices.

We drop our assumption of the symmetry of A and consider LU factorizations where L is unit lower triangular and U is upper triangular. The usual Gauss elimination algorithm for tridiagonal matrices is inherently serial. For example, if the i^{th} row of A is $(0, \dots, 0, c_i, a_i, b_i, 0, \dots, 0)$ the i^{th} element of the diagonal of U is given by the recursion formula

$$u_i = a_i - c_i b_{i-1} / u_{i-1}.$$

Since u_i depends on u_{i-1} , this cannot be directly computed using vector operations.

In Stone [1973a] it was noted that the recurrences of Gauss elimination could be reformulated into products of two \times two matrices and that these

products could be evaluated with vector operations using a technique known as recursive doubling. Recursive doubling in the simplest case expresses the $2i^{\text{th}}$ component of the product in terms of the i^{th} ; thus for $n = 2^k$ the n^{th} component can be computed in $\log_2 n = k$ steps.

As discussed in Lambiotte and Voigt [1975], this method is suitable for a computer such as ILLIAC IV but not for all other vector computers because for them each arithmetic operation costs some unit of time, and Stone's algorithm requires $O(n \log_2 n)$ arithmetic operations as opposed to $O(n)$ for the usual scalar algorithm. Thus, in the terminology of Lambiotte and Voigt [1975], Stone's algorithm is inconsistent.

In Stone [1975] and Lambiotte and Voigt [1975], various methods were analyzed, the latter concentrating on implementations for the STAR-100. The conclusion of both of these studies was that cyclic reduction was the best method for vector computers for matrices of sufficient size, say $n > 150$. Cyclic reduction was originally proposed by G. Golub and R. Hockney and is discussed in Hockney [1965] for tridiagonal systems arising from the five-point star for Poisson's equation. Subsequently, several authors including Hockney [1970] and Ericksen [1972] pointed out that the algorithm could also be adapted to general tridiagonal systems. The idea is to eliminate the odd numbered variables in the even numbered equations by performing elementary row operations. The details may be found in Lambiotte and Voigt [1975]; the important points are that the operations may be performed on vectors defined by the diagonals of the matrix, and that the resulting system is again tridiagonal but only half as large. The process may be continued until, in the case that $n = 2^k - 1$, only one equation remains; then all of the unknowns are recovered in a back substitution process. Lambiotte and Voigt [1975] show that this process requires only $O(n)$ operations and is thus consistent, but implementation

of the algorithm on vector computers can suffer from data rearrangement overhead that is as costly as the arithmetic operations. For example, on the STAR-100 one cannot operate with every other element of a vector as on the ASC (see Boris [1976]); thus extra operations must be employed to reformat those elements into a new vector. By considering the appropriate timing formulas it is easy to see that this overhead accounts for approximately half of the total operations. The overhead, as well as start up time, cause the algorithm to run slower on the STAR-100 than a carefully coded scalar version for small values of n . This is reflected in Table 1 which also includes times on a CDC CYBER-175* using the FTN compiler.

n	STAR-100 Gauss Elimination assembly language	STAR-100 Cyclic Reduction assembly language	CDC CYBER-175 Gauss Elimination FTN (opt. level 1)
50	560 μ sec	1039 μ sec	380 μ sec
100	1075	1337	720
150	1590	1612	1100
200	2160	1732	1400
300	3141	2126	2100
500	5204	2527	3400
1000	11070	3690	6900

Table 1.

Another comparison is given in Madsen and Rodrigue [1976] where a CDC-7600 is compared with the STAR-100. The authors discuss a polyalgorithm for the STAR-100 in which cyclic reduction is used until the matrix is reduced in size to the point where it is more efficient to solve by Gaussian elimination. Using this polyalgorithm they show that the STAR-100 is superior to

* The functional units of a CYBER 175 are essentially the same speed as those of a CDC 7600 but the memory is slower.

the 7600 only after $n > 750$. Their work also includes a comparison with an inconsistent algorithm proposed by Jordan [1974]; as expected it is slower than the Gaussian elimination-cyclic reduction polyalgorithm. Recently Swarztrauber [1976] has proposed a consistent tridiagonal scheme based on Cramer's rule. The operation count is slightly higher than that of cyclic-reduction but, as opposed to cyclic reduction, the algorithm is well defined mathematically for general non-singular tridiagonal matrices. Thus it might have merit in those situations where pivoting would be required for the stability of cyclic reduction. However, a stability analysis of Swarztrauber's algorithm has yet to be done.

A stable tridiagonal solver based on Given's transformations has been given by Sameh and Kuck [1976] for a theoretical parallel computer. The algorithm is inconsistent but slightly more efficient than the one given by Stone [1973a]. The stability analysis, which is contained in Sameh and Kuck [1977], is based on the recurrence relations that arise in the algorithm and the same approach may be used to show that Stone's algorithm is, in general, unstable (see Dubois and Rodrigue [1976]). Thus, for the ILLIAC IV or in the situation where the stability of cyclic reduction cannot be guaranteed, the algorithm based on Given's transformation may have merit.

Because of their inherent parallelism, iterative methods have been considered by Traub [1974] for solving tridiagonal problems, and further studied by Lambiotte and Voigt [1975] and Heller, Stevenson, and Traub [1976]. Except for certain specialized situations where an excellent starting value need only be improved by a few digits, these methods do not appear to be competitive with direct methods such as cyclic reduction.

There are, of course, matrices of interest whose bandwidth is too small for efficient use of banded solvers on vector computers, but is more than

one. There appear to be no good algorithms available for this problem. Rodrigue, Madsen, and Karush [1976] have proposed an extension of cyclic reduction to matrices of arbitrary bandwidth where the vector operations are on the diagonals as in cyclic reduction for tridiagonal matrices. Unfortunately, the numerical stability of the algorithm remains in doubt; indeed, even reasonable conditions on the banded matrix that guarantee that the algorithm remains well defined (i.e. division by zero cannot occur) have not yet been given.

We now turn our attention briefly to fast Poisson solvers based on the fast Fourier transform of Cooley and Tukey [1965]. Buzbee [1973] shows that for the five-point difference approximation to Poisson's equation on an M by N rectangular grid, ($M < N$), a technique known as matrix decomposition reduces the block tridiagonal system to M independent tridiagonal systems. The reduction to tridiagonal systems is accomplished via the fast Fourier transform and the systems may be solved in parallel on any of the computers discussed here.

As noted by Pease [1968], the fast Fourier transform may be computed efficiently on parallel computers. His algorithm was developed for an array computer with an interconnection pattern known as the perfect shuffle (see Stone [1971]), but Korn and Lambiotte [1977] have shown that it is also effective on the STAR-100. Ackins [1968] and Stevens [1971] developed a fast Fourier transform for the ILLIAC IV following the Cooley-Tukey approach and a similar algorithm is compared with the Pease algorithm on the STAR-100 in Korn and Lambiotte [1977].

Finally, Sameh, Chen, and Kuck [1976] treat Poisson's equation in a similar fashion to Buzbee [1973] on a theoretical array computer with a large number of independent processors. They also discuss the biharmonic equation, $\nabla^4 u(x,y) = F(x,y)$, and show that techniques used in the Poisson solver result in attractive algorithms for this equation.

4. Iterative and Time Marching Methods

We turn now to iterative methods for elliptic equations as well as methods for parabolic and hyperbolic equations. The reason for treating these together is that many of the considerations for both iterative as well as time-marching methods on vector computers are very similar, if not identical. Explicit methods will tend to be relatively more attractive than on serial computers because of their usually better vectorization properties, but this will not necessarily overcome the stringent stability requirements of small time steps. We shall illustrate this shortly.

The question of implicit versus explicit methods, however, is only one part of the broader consideration of how much of the method can be implemented by operations on vectors of a "good" length. Other aspects which affect this will include the domain, the boundary conditions (and perhaps computational boundary conditions needed for hyperbolic equations and/or higher order methods), the form of the coefficients and whether their calculation can be vectorized, the number of space dimensions, etc.

Parabolic and Hyperbolic Equations in One Dimension

We will begin the discussion with the simple parabolic equation

$$(4.1) \quad u_t = a u_{xx} \quad , \quad t > 0 \quad , \quad 0 < x < 1$$

with constant coefficient a and initial-boundary conditions

$$(4.2) \quad u(0,x) = g(x)$$

$$(4.3) \quad u(t,0) = \alpha \quad , \quad u(t,1) = \beta$$

for constant α and β .

Consider first the standard second-order Crank-Nicolson scheme

$$(4.4) \quad u_j^{k+1} - u_j^k = \frac{\mu}{2} (u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1} + u_{j+1}^k - 2u_j^k + u_{j-1}^k), \quad j = 1, \dots, N$$

where

$$(4.5) \quad \mu = a \frac{\Delta t}{(\Delta x)^2},$$

and u_j^k and u_j^{k+1} indicate values at the current and next time levels, respectively. At each time step, a tridiagonal system of equations must be solved and, as we saw in the last section, this is not particularly efficient on a vector computer with the algorithms now known. By contrast, the simplest explicit method

$$(4.6) \quad u_j^{k+1} = u_j^k + \mu(u_{j+1}^k - 2u_j^k + u_{j-1}^k), \quad j = 1, \dots, N,$$

is mechanistically ideal for vector computers, being carried out by 5 operations on vectors whose length is the number of interior grid points. Table 2 gives representative CPU times per step for (4.6) for the CDC STAR-100 and, for comparison, for the CDC CYBER 175. These times show that the optimal speed of 50 MFLOPS is almost attained for vectors of length $N = 1000$ but only a fraction of this for $N = 50$; in the latter case, almost three-fourths of the time for each vector operation is in start-up while for $N = 1000$, this penalty drops to about $\frac{1}{8}$. Table 2 also gives the CPU time for a Crank-Nicolson step, solving the tridiagonal systems by scalar Gaussian elimination for $N = 50$ and by odd-even reduction for

$N = 1000$. We see that the time ratio per step of the Crank-Nicolson and explicit methods is about 20 for $N = 1000$ and 14 for $N = 50$. On the other hand, the stability requirements for the explicit method are

$$\Delta t \leq \frac{2 \cdot 10^{-4}}{a} \quad \text{for } N = 50, \quad \Delta t \leq \frac{10^{-6}}{2a} \quad \text{for } N = 1000$$

which, depending on the accuracy requirements, may well cancel out the large per step time difference in favor of the explicit method.

	STAR	175
Explicit (Equation (4.6))		
$N = 50$	43 μsec	150 μsec
$N = 1000$	194 μsec	2500 μsec
Crank-Nicolson (4.4)*		
$N = 50$	600 μsec	560 μsec
$N = 1000$	3900 μsec	11700 μsec
Dufort-Frankel (4.7)		
$N = 50$	42 μsec	165 μsec
$N = 1000$	196 μsec	2800 μsec

Table 2

Time per Iteration for Various Methods

The Jacobi iteration for a tridiagonal system has a form similar to (4.6) and could be applied as an interior iteration for the Crank-Nicolson

*The tridiagonal systems were solved completely, rather than saving the L , U factors, at each time step since it was felt that this was more representative of realistic problems. Time for $N = 50$ is using Gaussian elimination; time for $N = 1000$ is using odd-even reduction.

or another implicit method. McCulley and Zaher [1974] have reported reasonable results with this approach for a diffusion problem on the ILLIAC IV; in their case, 15 Jacobi sweeps sufficed at each time-step.

The issue of explicit versus implicit methods points us in the direction of desiring an unconditionally stable explicit method. Perhaps the simplest such method is that of Dufort-Frankel (see, e.g. Richtmyer-Morton [1967]) which, for the problem (4.1)-(4.3), takes the form

$$u_j^{k+1} = u_j^{k-1} + 2\mu(u_{j+1}^k - u_j^{k+1} - u_j^{k-1} + u_{j-1}^k) \quad , \quad j = 1, \dots, N$$

or

$$(4.7) \quad u_j^{k+1} = \frac{1-2\mu}{1+2\mu} u_j^{k-1} + \frac{2\mu}{1+2\mu} (u_{j+1}^k + u_{j-1}^k)$$

This requires only 4 vector operations per time step although, since it is a two-level scheme, additional storage for u^{k-1} is required. Also, this scheme is inconsistent with (4.1) and care must be taken in the proper choice of $\Delta t/\Delta x$ to obtain a suitable approximation to the time-dependent solution. Representative times per iteration for this method are also given in Table 2. (The fact that the times are essentially identical to those for the explicit method (4.6), even though one less vector arithmetic operation is required, is due to needing two vector transmit instructions instead of one to set the current values in the correct arrays before the next time step. Note that one of these transmits could be avoided by interchanging the roles of u^k and u^{k-1} at each time step.)

In any of the above methods, the boundary conditions (4.3) are handled trivially by a one time insertion into the first and last positions of the vector which holds the approximation at the current time level. The initial

condition is also a one time calculation and can sometimes itself be vectorized. For example, if g is given by an explicit formula such as

$$(4.8) \quad g(x) = C_1 x + C_2 x^2$$

then the calculation of the initial vector can be done by the vector operations

$$(4.9) \quad C_1 * \underline{x} + C_2 * \underline{x} * \underline{x}$$

where $\underline{x} = (x_1, \dots, x_N)$ is the vector of grid points. Here, $\underline{x} * \underline{x}$ indicates the component by component product which is a hardware instruction on all current vector machines and takes essentially the same time as a scalar-vector multiplication.

So far, we have considered the most favorable circumstances: constant coefficients, constant boundary conditions, and one space dimension. Suppose, now, that the coefficient a in (4.1) is a function of x . Then μ in, for example, (4.6) is a function of the grid points and the implementation of (4.6) requires a component by component multiplication of the vector

$$(4.10) \quad \underline{\mu} = \frac{\Delta t}{\Delta x^2} (a(x_1), \dots, a(x_N))$$

with the second difference vector of u . Thus, if storage is available, the fact that a is a function of x adds only a one time evaluation of the vector $\underline{\mu}$ and this itself may perhaps be vectorized in the manner indicated for the initial condition. If a were also a function of t , then $\underline{\mu}$ would have to be recomputed at each time step and the ability to do this by vector operations would become much more important.

If the boundary conditions (4.3) were functions of t they would have to be computed at each time step, presumably by scalar operations and this would add an additional inefficiency to the code.

Let us turn briefly to hyperbolic equations in one space variable. Except for certain "stiff" hyperbolic systems (i.e. systems with a wide range of eigenfrequencies and characteristic phase velocities) implicit methods are rarely used, even on serial computers, and we will limit our attention to explicit methods.

The vectorization of explicit methods follows quite closely that for parabolic equations. Consider, for example, the hyperbolic system

$$(4.11) \quad \underline{u}_t + F(\underline{u})_x = 0, \quad 0 \leq x \leq 1$$

with initial condition

$$\underline{u}(0, x) = g(x) \quad 0 \leq x \leq 1$$

and suitable boundary conditions. The standard two-step Lax-Wendroff scheme (see, e.g. Richtmyer and Morton [1967]) is

$$(4.12) \quad \begin{aligned} \underline{u}_{j+\frac{1}{2}}^{k+\frac{1}{2}} &= \frac{1}{2}(\underline{u}_{j+1}^k + \underline{u}_j^k) - \gamma_1(F_{j+1}^k - F_j^k), \\ \underline{u}_j^{k+1} &= \underline{u}_j^k - \gamma_2(F_{j+\frac{1}{2}}^{k+\frac{1}{2}} - F_{j-\frac{1}{2}}^{k+\frac{1}{2}}) \end{aligned}$$

where $\gamma_2 = \Delta t / \Delta x$ and $\gamma_1 = \gamma_2 / 2$.

Assume that there are N equations and M interior grid points. The simplest approach is to treat each component of the system (4.11) separately in (4.12) and vectorize over the number of grid points; that is, let U_1, \dots, U_N , U_1, \dots, U_N , and F_1, \dots, F_N be one dimensional arrays of length $M + 2$ and use the notation $U(I; J)$ to denote the subarray of length J starting in

position I . Then the first part of (4.12) can be carried out by the vector instructions

$$(4.13) \quad UT1(1;M+1) = .5*(U1(2;M+1) + U1(1;M+1)) - \gamma_1*(F1(2;M+1) - F1(1;M+1))$$

and similarly for $UT2, \dots, UTN$. Next we need to evaluate F at the points of $UT1, \dots, UTN$ and in many cases this can be done primarily also by vector instructions. For example, suppose that u is the 3-vector of density ρ , momentum m , and energy e and $F = (m, p + m^2/\rho, (e + p)m/\rho)$ where p is given in terms of ρ , and possibly also m and e , by some "equation of state"

$$p = f(\rho, m, e)$$

Then the evaluation of F can be done by vector operations as indicated in

$$F_j = (m_j, p_j + \frac{m_j^2}{\rho_j}, (e_j + p_j) \frac{m_j}{\rho_j}) \quad , \quad j = 1, \dots, N$$

where the calculation of the vector of p values may or may not also be computed efficiently by vector operations depending on the form of f .

After F has been evaluated, vector instructions similar to (4.13) can be used for the second step of (4.12). In addition, we will need to handle the given boundary conditions as well as the computational boundary conditions obtained, for example, by extrapolation (see e.g. Gottlieb and Turkel [1976] for a review and analysis of various extrapolation strategies).

The above approach could be made more efficient if vectors of length MN could be used rather than just of length M . In some cases, this will be possible using some techniques that we discuss next for elliptic equations.

Elliptic Equations in Two Dimensions

The implementation of many of the the usual iterative methods for discrete elliptic equations has been studied rather extensively by Erickson [1972], Hayes [1974], and Lambiotte [1975], primarily for the ILLIAC IV, the ASC, and the STAR-100, respectively, and Morice [1972] for general parallel processors. We will review in this section many of the issues they have discussed and, for simplicity, refer to their work collectively by EHLM unless an individual reference is more appropriate. We also note here the paper by Heller [1977], which surveys many aspects of iterative methods for parallel computers.

The detailed considerations of EHLM were primarily restricted to the model problem of Poisson's equation on a square with Dirichlet boundary data and using the five point difference star, and this will be our starting point also. Such a problem would, of course, actually be solved by one of the fast Poisson solvers mentioned in the last section but it makes a convenient example with which to treat many of the issues of vectorization that arise in more general problems.

The discrete domain is indicated in Figure 5 where the boundary points are indicated by bold dots and N is the number of interior points in each row and column.

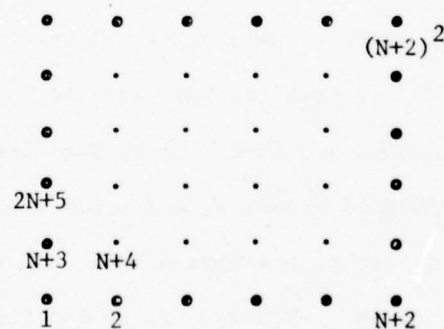


Figure 5

Consider the Fortran loop

```

      DO 1 J = 2, N + 1
(4.14) DO 1 I = 2, N + 1
      1 UN(I,J) = .25*(U(I+1,J)+U(I-1,J)+U(I,J+1)+U(I,J-1))

```

where the current values of the unknowns as well as the boundary values are stored in an $(N+2) \times (N+2)$ array. Equation (4.14) defines the arithmetic step of the Jacobi iteration, a process which (for general linear systems as well as the discrete laplacian represented by (4.14)) has been extensively cited as a prototype parallel method. However, care is needed in certain aspects of its implementation in order to achieve the fullest possible vectorization. For example, (4.14) would be more efficiently implemented on the ILLIAC in a row by row fashion if N were 64 or a multiple thereof and an inefficiency of up to a factor of 2 could result for other values. On the STAR and ASC, on the other hand, we would like the vector lengths to be as long as possible.

For any reasonable value of N (4.14) translates directly into 4 hardware instructions on the ASC which operate on vectors of effective length $O(N^2)$. On the STAR, this is not possible since a vector consists only of contiguous memory locations. One could carry out (4.14) row by row but then the vectors would only be of length N and one would pay N times the number of start-up penalties. Alternatively, we can use vectors of length $O(N^2)$ by treating the boundary positions as unknowns. That is, let U now denote an $(N+2)^2$ long one-dimensional array with the lexicographic ordering of Figure 5, and again use the notation $U(K;L)$ to denote the L -long subvector starting at the K th position of U . With $M1 = (N+1)(N+2) - 1$ and $M2 = N(N+2) - 2$, we can then implement (4.14) by the instructions

$$\begin{aligned}
 (4.15) \quad T(2;M1) &= U(2;M1) \frac{+}{2} U(N+3;M1) \\
 U(N+4;M2) &= T(2;M2) \frac{+}{2} T(N+5;M2)
 \end{aligned}$$

where T is a temporary vector and where we have used $\frac{+}{2}$ to denote an "average" instruction, that is, addition followed by division by 2. Such an instruction is available on the STAR and takes essentially the same time as an addition.

As a penalty for using vectors whose length is the total number of grid points, the final instruction of (4.15) will write on to the positions

$$2N+4, 2N+5, 3N+6, 3N+7, \dots$$

corresponding to most of the boundary positions along the vertical sides, thus destroying the correct boundary values in those positions. One would then have to restore these values before the next iteration. On the STAR, however, there is a convenient feature which permits storage to be controlled by a bit vector (the control vector); this can be used to ensure that the boundary positions are not overwritten and, hence, no "fixing up" is needed before the next iteration. Let B be a bit vector with 0's in those positions corresponding to the boundary points in the vector $U(N+4;M2)$ and 1's in the interior grid point positions. The last instruction of (4.15) would then be replaced by

$$(4.16) \quad U(N+4;M2) \cdot \text{Control } B = T(2;M2) \frac{+}{2} T(N+5;M2)$$

indicating that storage is suppressed into those positions of $U(N+4;M2)$ corresponding to 0's in B i.e., corresponding to the boundary positions. The instruction time is no greater using the control vector but, of course, one pays the penalty of storage of approximately $N^2/64$ words of storage for the bit vector.

The temporary vector in (4.15) can be eliminated by storing the result of the first instruction back onto U itself. However, even using a control vector, this will necessarily destroy about half of the boundary values which will then have to be replaced.

Whereas the Jacobi iteration is often cited as a "perfect" parallel algorithm, the Gauss-Seidel and SOR iterations are usually considered to be the opposite. The usual serial code for Gauss-Seidel in the context of the DO loop (4.14) would have UN replaced by U itself so that new values at each point replace the old as soon as they are computed; it is this process that is not amenable to vectorization. However, EHLM have shown that by using the classical red-black ordering of the grid points, as shown in Figure 6,

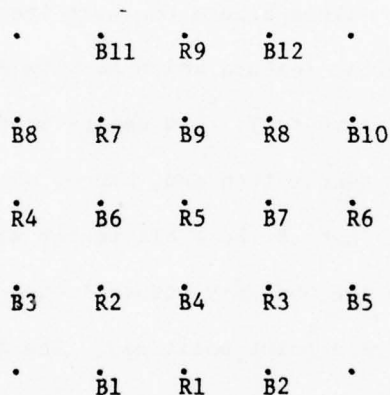


Figure 6. The Red-Black Ordering

Gauss-Seidel can be carried out in essentially the same fashion as the Jacobi iteration but using vectors of length $O(\frac{N^2}{2})$ corresponding to the red points and the black points. The boundary points would be handled in the same way as with Jacobi's method. The time per iteration for SOR carried out in this way

should be little more than for the Jacobi iteration so that the SOR method is potentially very useful for vector machines. Lambiotte [1975] also considered a diagonal ordering for the grid points but showed that this is inferior to the red-black ordering.

Similarly to SOR, the Alternating Direction Implicit (ADI) method seems, at first glance, to be rather unsatisfactory for vector computers since it is based on the solution of tridiagonal equations. Erickson [1972] and Morice [1972], however, observed that since these tridiagonal systems are independent, they can be solved in parallel. More precisely, we recall that the ADI method - for the model problem and the grid of Figure 5 - consists of two half-steps as indicated by the iteration scheme (see, e.g. Varga [1962])

$$(4.17a) \quad (H + \alpha_k I) \underline{x}^{k+\frac{1}{2}} = (\alpha_k I - V) \underline{x}^k + \underline{b}$$

$$(4.17b) \quad (V + \alpha_k I) \underline{x}^{k+1} = (\alpha_k I - H) \underline{x}^{k+\frac{1}{2}} + \underline{b}$$

The first step, (4.17a), consists of the solution of N tridiagonal systems corresponding to the horizontal lines of the grid, while (4.17b) likewise is the solution of N tridiagonal systems (after permutations of the unknowns) corresponding to the vertical lines. On the ILLIAC, up to 64 of these tridiagonal systems can be solved in parallel using the usual Gaussian elimination algorithm. Moreover, the storage mechanism on the ILLIAC is such that the alternating sweeps can be handled with little difficulty.

On the STAR or ASC, the vectorization would be across the tridiagonal systems; that is, suppose, more generally, that we have N $n \times n$ linear systems

$$A^{(k)} \underline{x}^{(k)} = \underline{b}^{(k)}, \quad k = 1, \dots, N$$

with $A^{(k)} = (a_{ij}^{(k)})$ and $\underline{b}^{(k)} = (b_i^{(k)})$. We then set up $n^2 + n$ vectors

$$(4.18) \quad \begin{aligned} A_{ij} &= (a_{ij}^1, \dots, a_{ij}^N) & i, j &= 1, \dots, n \\ B_i &= (b_i^1, \dots, b_i^N) & i &= 1, \dots, n \end{aligned}$$

and solve the N systems by carrying out Gaussian elimination in the usual fashion, but now on the vectors of (4.18).

In order to implement this for the tridiagonal systems in the ADI method, it is primarily a question of arranging the storage layout correctly. On the ASC this can be done with little difficulty, and with essentially the usual serial Fortran code, because of that machine's ability to treat non-contiguous elements as vectors. On the STAR, however, the storage must be rearranged - by the equivalent of a matrix transpose - between each sweep. Lambiotte [1975] suggests the following alternative: On the half-sweep that the storage is not correct for the simultaneous solution of the tridiagonal systems, it is correct for the solution of the individual tridiagonal systems by the cyclic reduction (CR) method of the previous section. Moreover, the N individual systems may be viewed as forming a single tridiagonal system N times as large and the CR method may be applied to this large system; as we saw in the last section, the larger the system the better. Finally, because the individual systems are uncoupled, the CR method will actually terminate in $\log_2 N$ steps rather than the expected $\log_2 N^2$. Thus, the ADI algorithm is implemented by solving the tridiagonal systems "in parallel" on one half-sweep and as a single large tridiagonal system on the other half-sweep. Lambiotte also discusses a similar strategy for three dimensional problems.

In a limited number of recent experiments on the STAR, however, Lambiotte [1977] has shown that it actually seems to make little difference in running time whether the above ADI algorithm is implemented or the tridiagonal systems solved in parallel on each half-sweep with a matrix transpose in between. This is presumably due to the fact that there is a hardware 8x8 matrix transpose instruction on the STAR which forms the basis for an efficient transpose procedure.

Many of the same considerations as for ADI apply also to the implementation of Successive Line Over-Relaxation (SLOR). If one uses the lexicographic ordering, the same difficulties occur as with point SOR. To circumvent this, Ericksen [1972] and Lambiotte [1975] study various other orderings, such as the red-black ordering by lines, which allow a number of the tridiagonal systems either to be solved in parallel or as one large tridiagonal system by cyclic reduction.

Another group of methods which may be potentially useful on vector machines is the semi-iterative (SI) methods. (See e.g. Young [1971] for a general discussion of these methods.) Consider, for example, the Jacobi-SI method which can be written in the form

$$(4.19) \quad \underline{u}^{k+1} = \alpha_k B \underline{u}^k + \beta_k \underline{u}^k + \gamma_k \underline{u}^{k-1}$$

for suitable choice of the parameters α, β , and γ . Here B is the Jacobi iteration matrix so that $B \underline{u}^k$ is the result of a Jacobi sweep starting from \underline{u}^k , and the remainder of the calculation of (4.19), once the parameters are known, is ideally suited for vector machines. Of course, one pays the penalty of additional storage for \underline{u}^{k-1} . More importantly, the choice of good parameters may be difficult for other than the model problem: the optimal parameters are based on a knowledge of the largest and smallest eigenvalues

(assumed real) of B . In the case that the coefficient matrix A is symmetric positive definite and has property A , then it is known that the asymptotic rate of convergence of Jacobi-SI is approximately half that of SOR, both using optimal parameters; in this case, Jacobi-SI may not be useful, even on vector computers. However, in more general situations, the rate of convergence of Jacobi-SI may be quite superior to SOR and its somewhat better vectorization properties makes it potentially attractive, provided that reasonable values of the parameters can be chosen. Hayes [1974] and Lambiotte [1975] have considered, for the ASC and STAR, respectively, the Jacobi-SI method in some detail, as well as other semi-iterative methods such as SSOR-SI and cyclic Chebyshev-SI.

So far we have considered explicitly only the model problem of Poisson's equation on a square, although, of course, parts of the previous discussion apply more generally. With a more general elliptic operator - but still a rectangular region - the difficulties will tend to revolve around the best ways to compute and manage storage of the coefficients. For example, consider the equation

$$(4.20) \quad a u_{xx} + b u_{yy} + c u_{zz} = f, \quad 0 \leq x, y, z \leq 1$$

where a , b , and c are functions of x , y , and z . The corresponding difference equations using the usual 7-pt formula with $h = \Delta x = \Delta y = \Delta z$ are

$$\begin{aligned} & 2(a_{ijk} + b_{ijk} + c_{ijk})u_{ijk} - a_{ijk}(u_{i+1,j,k} + u_{i-1,j,k}) \\ & - b_{ijk}(u_{i,j+1,k} + u_{i,j-1,k}) - c_{ijk}(u_{i,j,k+1} + u_{i,j,k-1}) = h^2 f_{ijk} \end{aligned}$$

For a sufficiently coarse grid, the coefficients can all be computed once and for all and held in five $O(N^3)$ long arrays. But for a moderately fine

grid, say $N = 50-100$, out-of-core storage would be required and depending upon the complexity of the coefficients, the operating system, and various other factors, it may be more economical to recompute the coefficients at each iteration. This strategy is, of course, common on existing serial machines and the only new factor for vector computers would be to compute the coefficients in sufficiently large batches - say 1,000 - 10,000 at a time - so that the computation as well as the subsequent usage in the iterative scheme can be done efficiently with vector operations.

A less satisfactory situation exists for handling irregular domains. Consider, for example, the grid in Figure 7

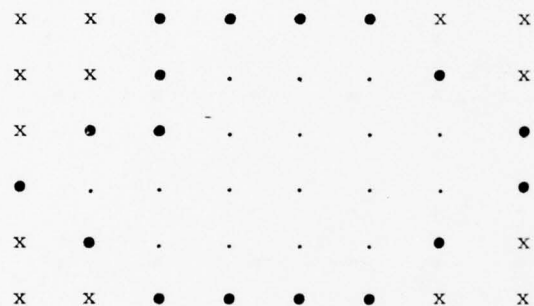


Figure 7

where the boundary nodes are indicated by bold dots. One way to handle such a grid is to circumscribe it by a rectangle - the additional grid points thus introduced are indicated in Figure 7 by crosses - and work with the entire rectangular grid. For example, for Laplace's equation and the Jacobi iteration on such a grid, one could use the DO loop (4.14) on the ASC or the vector code (4.15) on the STAR. For the ASC, the values at the boundary nodes in the interior of the rectangle will be destroyed and must be replaced prior to the next sweep. On the STAR, a control vector can be used, as before, to

ensure that the boundary positions are not overwritten. Of course, both additional storage as well as additional arithmetic are required for the rectangular points outside of the domain and the procedure becomes increasingly less efficient as the domain deviates from a rectangle. At some point, it is probably beneficial to use a union of smaller circumscribing rectangles, as indicated in Figure 8,

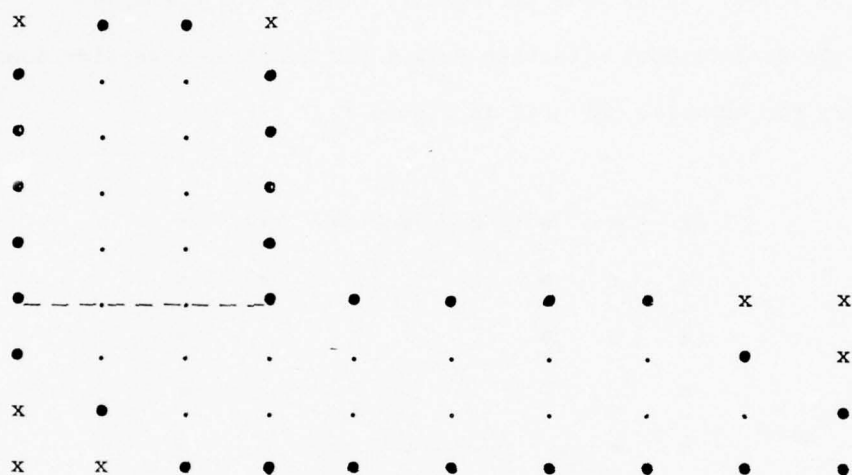


Figure 8.

for the case of two such rectangles. This, of course, would save considerable storage over a complete circumscribing rectangle but now the two rectangles must be processed separately. That is, the DO loop (4.14) or the code (4.15) must be written separately for the two rectangles to take account of the different row lengths as the dotted line is crossed. Ideally, of course, one would like an ordering of the grid points that would allow processing and storage of only the minimum number of points and still use vectors whose length is the total number of grid points; but such an ordering, if it exists, is not evident and has not appeared in the literature.

One might argue that the above discussion on iterative methods for elliptic equations using finite difference discretizations is somewhat irrelevant since the trend of the last several years has been, increasingly, to use projection methods (i.e. finite elements, Galerkin, etc.) to discretize elliptic equations and then direct methods for the solution of the algebraic systems. We note several points about this. First, finite differences have been used here primarily to simplify the discussion but many of the same considerations would apply to the discussion of iterative methods for finite element or other projection discretizations. Secondly, iterative methods seem to vectorize somewhat better, on the whole, than direct methods and the balance of favor may well shift towards iterative methods for vector computers, especially for three dimensional problems. Thirdly, as mentioned previously, many of the considerations for handling iterative methods go over directly to time-marching methods for initial-boundary value problems. Finally, iterative methods play a prominent role in the "multigrid" method, one of the currently most promising approaches to the solution of elliptic equations. (See Brandt [1977] and Nicolaides [1975] for discussion of this method.)

5. Summary and Prognosis

The use of large scale vector computers in scientific computation is barely four years old and the serious study of the best ways to use such computers has been largely restricted to the half dozen or so government laboratories where such computers exist or where active preparation for one is under way. The university community of numerical analysts and others in scientific computing has been rather little involved to date in developing or analyzing numerical methods for such computers. Moreover, the existing machines are sufficiently different in detail that it is not always the case ~~that~~ an algorithm which is best for one machine will even be satisfactory for another.

Nevertheless, progress has been made. The implementation of the usual direct methods for solving linear systems of equations is now well understood although not altogether satisfactory: Gaussian elimination or similar factorization methods become relatively less efficient as the bandwidth decreases; in the important limit of tridiagonal matrices, cyclic reduction appears to be the best method although for matrices of moderate size ($n = 100-500$) it is relatively little, if at all, better than Gaussian elimination in scalar code.

Most of the usual iterative methods have now been studied to some extent and, in general, seem somewhat more competitive on vector computers (vis-a-vis direct methods) than on serial ones. But most analysis to date has been on simple model problems and many questions remain for more realistic situations: irregular domains, systems of equations, variable coefficients, etc. And some of the most potentially promising iterations, such as the conjugate gradient method or the multigrid method, have not yet been seriously studied.

For time-dependent initial-boundary value problems the implementation of explicit methods has much in common with that of iterative methods for elliptic equations. But little analysis or comparative study has yet been carried out although various methods are being used. The issue of explicit versus implicit methods for parabolic equations is still unclear although explicit methods seem relatively more advantageous on vector computers. The method of lines — which is becoming increasingly popular as a basis for "automatic" programs — has not yet been seriously studied.

All of the existing vector computers have both strong and weak points. Ideally, we would like machines that allow maximum flexibility in the machine definition of a vector — at a minimum, equally spaced storage words should be allowed; minimum overhead (start-up time) for vector hardware operations; no penalty, relative to the fastest available technology, for the use of scalar arithmetic; sufficient amounts of fast memory to balance the high CPU processing rates, and so on. On the software side, we especially need more sophisticated programming languages and compilers.

None of the existing vector computers have all of these desirable features but, then, we are only in the first generation of such machines. Moreover, we seem to be witnessing a genuine revolution in hardware and increasingly serious speculation is being made about array processors of the future which will utilize thousands of independent microprocessors. (For a prognosis of computer power in the 1980's, see Stone [1976].) The knowledge we are slowly gaining today about the efficient use and the drawbacks of the current vector computers will surely be useful for the next generation of such computers.

Acknowledgement We are grateful to T. Craig Poling for expert programming assistance, and to J. Boris, D. Calahan, D. Heller, J. Lambiotte, D. Stevenson, and H. Stone for valuable comments.

References

- Ackins, G [1968]. Fast Fourier Transform via ILLIAC IV. ILLIAC IV Document No. 168, University of Illinois at Urbana - Champaign.
- Barnes, G., Brown, R., Kato, M., Kuck, D., Slotnick, D., and Stokes, R. [1968]. The ILLIAC IV Computer. IEEE Trans. Computers C-17, pp. 746-757.
- Berg, R., et al. [1972]. PEPE - An Overview of Architecture, Operation and Implementation. Proc. National Electronics Conference, IEEE, New York, pp. 312-317.
- Boris, J. [1976]. Flux-Corrected Transport Modules for Solving Generalized Continuity Equations. NRL Memorandum Report 3237. Naval Research Laboratory, Washington, D.C.
- Boris, J. [1976]. Vectorized Tridiagonal Solvers. NRL Memorandum Report 3408. Naval Research Laboratory, Washington, D.C.
- Bouknight, W., Denenberg, S., McIntyre, D., Randall, J., Sameh, A. and Slotnick, D. [1972]. The ILLIAC IV System. Proc. IEEE 60, pp. 369-388.
- Brailovskaya, I. [1965]. A Difference Scheme for Numerical Solution of the Two-Dimensional Non-stationary Navier-Stokes Equations for a Compressible Gas. Soviet Physics Doklady 10, pp. 107-110.
- Brandt, A. [1977]. Multi-Level Adaptive Solutions to Boundary Value Problems. To appear in Mathematics of Computation.
- Buzbee, B. [1973]. A Fast Poisson Solver Amenable to Parallel Computation. IEEE Trans. Computers C-22, pp. 793-796.
- Calahan, D. [1975]. Complexity of Vectorized Solution of Two-Dimensional Finite Element Grids. SEL Report No. 91, Systems Engineering Laboratory, University of Michigan.
- Calahan, D., Joy, W. and Orbits, D. [1976]. Preliminary Report on Results of Matrix Benchmarks on Vector Processors. SEL Report No. 94, Systems Engineering Laboratory, University of Michigan.
- Carroll, A. and Wetherald, R. [1967]. Application of Parallel Processing to Numerical Weather Prediction. J. Assoc. Comp. Mach. 14, pp. 591-614.
- Control Data Corporation [1975]. Control Data STAR-100 Computer Hardware Reference Manual, Rev. 9. Arden Hills, Minnesota.
- Cooley, J. and Tukey, J. [1965]. An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation 19, pp. 297-301.
- Cray Research Corporation [1976]. CRAY-1 Computer System Reference Manual, Rev. A. Bloomington, Minnesota.
- Davy, W. and Reinhardt, W. [1975]. Computation of Shuttle Non-equilibrium Flow Fields on a Parallel Processor. NASA SP-347, pp. 1351-1376.

- Dubois, P. and Rodrigue, G. [1976]. An Analysis of the Recursive Doubling Algorithm. Lawrence Livermore Laboratory Preprint No. UCRL-79071, Livermore, California.
- Ericksen, J. [1972]. Iterative and Direct Methods for Solving Poisson's Equation and Their Adaptability to ILLIAC IV. CAC Document No. 60, Center for Advanced Computation, University of Illinois at Urbana - Champaign.
- Ericksen, J. and Wilhelmson, R. [1976]. Implementation of a Convective Problem Requiring Auxiliary Storage. ACM Trans. Math. Software 2, pp. 187-195.
- Fong, K. and Jordan, T. [1976]. Some Linear Algebraic Algorithms and Their Performance on CRAY-1. Report written under U. S. Energy Research and Development Administration Contract No. W-7405-ENG-36 at Los Alamos Scientific Laboratory, Los Alamos, New Mexico.
- George, A. [1972]. An Efficient Band-oriented Scheme for Solving n by n Grid Problems. Proc. 1972 FJCC, AFIPS Press, Montvale, New Jersey, pp. 1317-1321.
- George, A. [1973]. Nested Dissection on a Regular Finite Element Mesh. SIAM J. Numer. Anal. 10, pp. 345-363.
- George, A. [1977]. Numerical Experiments Using Dissection Methods to Solve n by n Grid Problems. SIAM J. Numer. Anal. To appear in Vol. 14, No. 2.
- George, A. and Liu, J. [1976]. An Automatic Nested Dissection Algorithm for Irregular Finite Element Problems. Research Report CS-76-38, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.
- George, A., Poole, W. and Voigt, R. [1976a]. A Variant of Nested Dissection for Solving n by n Grid Problems. Report No. 76-16, ICASE, NASA Langley Research Center, Hampton, Virginia.
- George, A., Poole, W. and Voigt, R. [1976b]. Analysis of Dissection Algorithms for Vector Computers. Report No. 76-17, ICASE, NASA Langley Research Center, Hampton, Virginia.
- Gilmore, P. [1971]. Numerical Solution of Partial Differential Equations by Associative Processing. Proc. 1971 FJCC AFIPS Press, Montvale, New Jersey, pp. 411-418.
- Goodyear Aerospace Corporation [1974]. System Description - STARAN. Akron, Ohio.
- Gottlieb, D. and Turkel, E. [1976]. Boundary Conditions for Multistep Finite Difference Methods for Time Dependent Equations. Report No. 76-30, ICASE, NASA Langley Research Center, Hampton, Virginia.

- Graham, M. [1976]. An Array Computer for the Class of Problems Typified by the General Circulation Model of the Atmosphere. Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana - Champaign.
- Graves, R. [1973]. Partial Implicitization. J. Computational Physics 13, pp. 439-444.
- Hayes, L. [1974]. Comparative Analysis of Iterative Techniques for Solving Laplace's Equation on the Unit Square on a Parallel Processor. M. S. Thesis, Department of Mathematics, University of Texas, Austin.
- Heller, D. [1977]. A Survey of Parallel Algorithms in Numerical Linear Algebra. SIAM Review, to appear.
- Heller, D., Stevenson, D. and Traub, J. [1976]. Accelerated Iterative Methods for the Solution of Tridiagonal Systems on Parallel Computers. J. Assoc. Comp. Mach. 23, pp. 636-654.
- Hockney, R. [1965]. A Fast Direct Solution of Poisson's Equation Using Fourier Analysis. J. Assoc. Comp. Mach. 12, pp. 95-113.
- Hockney, R. [1970]. The Potential Calculation and Some Applications. Methods Computational Physics 9, Academic Press, New York, pp. 135-211.
- Jennings, A. [1966]. A Compact Storage Scheme for the Solution of Symmetric Linear Simultaneous Equations. Comput. J. 9, pp. 281-285.
- Jordan, T. [1974]. A New Parallel Algorithm for Diagonally Dominant Tri-diagonal Matrices. Los Alamos Scientific Laboratory Report, Los Alamos, New Mexico.
- Keller, T. [1976]. Cray-1 Evaluation Final Report. Informal Report LA-6456-MS. Los Alamos Scientific Laboratory, Los Alamos, New Mexico.
- Knight, J., Poole, W. and Voigt, R. [1975]. System Balance Analysis for Vector Computers. Proc. 1975 ACM National Conference, pp. 163-168.
- Korn, D., and Lambiotte, J. [1977]. On Computing the FFT on the CDC STAR-100. To appear.
- Lambiotte, J. [1975]. The Solution of Linear Systems of Equations on a Vector Computer. Ph.D. Dissertation, University of Virginia.
- Lambiotte, J. [1977]. Private Communication.
- Lambiotte, J., and Howser, L. [1974]. Vectorization on the STAR Computer of Several Numerical Methods for a Fluid Flow Problem. NASA TND-7545, Langley Research Center, Hampton, Virginia.
- Lambiotte, J. and Voigt, R. [1975]. The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer. ACM Trans. Math. Software 1, pp. 308-329.
- Madsen, N. and Rodrigue, G. [1976]. A Comparison of Direct Methods for Tri-diagonal Systems on the CDC STAR-100. Preprint UCRL-76993, Rev. 1. Lawrence Livermore Laboratory, Livermore, California.

- MacCormack, R. and Stevens, K. [1976]. Fluid Dynamics Applications of the ILLIAC IV Computer. In "Computational Methods and Problems in Aeronautical Fluid Dynamics" Academic Press, New York, pp. 448-465.
- McCulley, L. and Zaher, G [1974]. Heat Shield Response to Conditions of Planetary Entry Computed on the ILLIAC IV. Unpublished manuscript under NASA/Ames Contract No. 6911
- McMahon, F., Sloan, L. and Long, G. [1972]. STACKLIB - A Vector Function Library of Optimum Stack-loops for the CDC 7600. Lawrence Livermore Laboratory Report, Livermore, California.
- Miranker, W. [1971]. A Survey of Parallelism in Numerical Analysis. SIAM Rev. 13, pp. 524-547.
- Morice, P. [1972]. Calcul Parallele et Decomposition Dans la Resolution d'equations Aux Derivees Partielles de Type Elliptique. IRIA, Rocquencourt, France.
- Nicolaides, R. [1975]. On Multiple Grid and Related Techniques for the Solution of Discrete Elliptic Systems. J. Computational Physics 19, pp. 418-431.
- Noor, A. and Fulton, R [1975]. Impact of the CDC-STAR-100 Computer on Finite-Element Systems. J. Structural Div. ASCE. 101, no. ST4, pp. 731-750.
- Noor, A. and Voigt, S. [1975]. Hypermatrix Scheme for Finite Element Systems on CDC STAR-100 Computer. J. Comput. and Structures 5, pp. 287-296.
- Pease, M. [1968]. An Adaptation of the Fast Fourier Transform for Parallel Processing. J. Assoc. Comp. Mach. 15, pp. 252-264.
- Poole, W. and Voigt, R. [1974]. Numerical Algorithms for Parallel and Vector Computers: An Annotated Bibliography. Computing Reviews 15, pp. 379-388.
- Ogura, M., Sher, M. and Ericksen, J [1972]. A Study of the Efficiency of ILLIAC IV in Hydrodynamic Calculations. CAC Document No. 59, Center for Advanced Computation, University of Illinois at Urbana - Champaign.
- Orbits, D. and Calahan, D. [1976]. Data Flow Considerations in Implementing a Full Matrix Solver with Backing Store on the CRAY-1. SEL Report No. 98, Systems Engineering Laboratory, University of Michigan.
- Ramamoorthy, C. and Li, H. [1977]. Pipeline Architecture. Computing Surveys 9, pp. 61-102.
- Reilly, B. [1970]. On Implementing the Monte Carlo Evaluation of the Boltzmann Collision Integral on ILLIAC IV. Report No. I-140, Coordinated Science Laboratory, University of Illinois at Urbana - Champaign.
- Richtmyer, R. and Morton, K. [1967]. Difference Methods for Initial-Value Problems. Interscience, New York.
- Rodrigue, G., Madsen, N., and Karush, J. [1976]. Odd-Even Reduction for Banded Linear Equations. Preprint UCRL-78652, Lawrence Livermore Laboratory, Livermore, California.

- Rudolph, J. [1972]. A Production Implementation of An Associative Array Processor - STARAN. Proc. 1972 FJCC AFIPS Press, Montvale, New Jersey, pp. 229-241.
- Sameh, A., Chen, S., and Kuck, D. [1976]. Parallel Poisson and Biharmonic Solvers. Computing 17, pp. 219-230.
- Sameh, A. and Kuck, D. [1976]. On Stable Parallel Linear System Solvers. Department of Computer Science Report, University of Illinois at Urbana - Champaign.
- Sameh, A. and Kuck, D. [1977]. A Parallel QR Algorithm for Symmetric Tridiagonal Matrices. IEEE Trans. Computers, C-26, pp. 147-153.
- Swarztrauber, P. [1976]. A Parallel Algorithm for Solving General Tridiagonal Equations. National Center for Atmospheric Research Report, Boulder, Colorado.
- Stevens, J. [1971]. A Fast Fourier Transform Subroutine for ILLIAC IV. CAC Document No. 17, Center for Advanced Computation, University of Illinois at Urbana - Champaign.
- Stone, H. [1971]. Parallel Processing with the Perfect Shuffle. IEEE Trans. Computers C-20, pp. 153-161.
- Stone, H. [1973a]. An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations. J. Assoc. Comp. Mach. 20, pp. 27-38.
- Stone, H. [1972b]. Problems of Parallel Computation. Complexity of Sequential and Parallel Numerical Algorithms, J. Traub, ed. Academic Press, New York pp. 1-16.
- Stone, H. [1975]. Parallel Tridiagonal Equation Solvers. ACM Trans. Math. Software 1, pp. 289-301.
- Stone, H. [1976]. Computer Architecture in the 1980s. Proc. of the Third ICASE Conference on Scientific Computing: Computer Science and Scientific Computing, J. Ortega ed. Academic Press, New York.
- Texas Instruments, Inc. [1973]. The ASC System - Central Processor. Austin, Texas.
- Traub, J. [1974]. Iterative Solution of Tridiagonal Systems on Parallel and Vector Computers. Complexity of Sequential and Parallel Numerical Algorithms, J. Traub ed. Academic Press, New York. pp. 49-82.
- Voigt, R. [1977]. The Influence of Vector Computer Architecture on Numerical Algorithms. Report No. 77-8, ICASE, NASA Langley Research Center, Hampton, Virginia. To appear in Proceedings of the Symposium on High Speed Computer and Algorithm Organization. D. Kuck ed. Academic Press, New York.

- von Fuchs, G., Roy, J. and Schrem, E. [1972]. Hypermatrix Solution of Large Sets of Symmetric Positive-Definite Linear Equations. Comput. Methods Appl. Mech. Engng. 1, pp. 197-216.
- Weilmunster, J. and Howser, L. [1976]. Solution of a Large Hydrodynamic Problem using the STAR-100 Computer. NASA TM X-73904, Langley Research Center, Hampton, Virginia.
- Wilhelmson, R. [1974]. Solving Partial Differential Equations using ILLIAC IV. In Constructive and Computational Methods for Differential and Integral Equations, A. Dold and B. Eckmann, eds., Springer-Verlag, New York, pp. 453-476.
- Wulf, W. and Bell, C. [1972]. C.mmp - A Multi-Mini-Processor. Proc. 1972 FJCC AFIPS Press, Montvale, New Jersey, pp. 765-777.
- Young, D. [1971]. Iterative Solution of Large Linear Systems. Academic Press, New York.

A MATHEMATICAL SOLUTION TO THE BALLISTIC DIFFERENTIAL EQUATION
PROGRAMMED FOR THE TEXAS INSTRUMENT SR-52 CALCULATOR

Thomas H. Slook
Temple University and Frankford Arsenal
Philadelphia, Pennsylvania 19137

I. ABSTRACT. The fundamental differential equation of motion for a projectile is

$$(1.1) \quad \dot{\vec{v}} + H\vec{v} - \vec{g} = \vec{0}$$

where \vec{v} is the projectile velocity, \vec{g} is the acceleration due to gravity and H is a function of the dependent variable v . If it were not for the fact that H is a complicated function of the magnitude of projectile velocity, then equation (1.1) would have a closed form solution. Specifically, H is defined by equation

$$(1.2) \quad H = \frac{\rho d^2 v K_D}{w}$$

where

d = diameter of projectile (meters),
 w = weight of projectile (kilograms),
 v = speed of projectile (meters/second),
 K_D = drag function and
 $\rho = \rho^* e^{-hz}$

with ρ^* = standard air density (kilograms/meters³), a = altitude of projectile (meters) and $h = 0.000045 \log_{10} \frac{a}{8}$ per meter. Observe that the unit of measure for H is a real number per second.

In this paper the fundamental ballistic equation is solved by an iterative method which generates a five degree of freedom program for the SR-52 with the following outputs: ① time of flight (seconds), ② range (meters), ③ altitude (meters), ④ remaining projectile speed (meters/sec), ⑤ range rate (meters/sec), ⑥ altitude rate (meters/sec) and ⑦ projectile angle of rise-fall (radians).

II. INTRODUCTION. The fact that there is no closed form solution to the non-linear ballistic equation indirectly lead to the development of the ENIAC in 1946 at the University of Pennsylvania. This very large vacuum tube machine was man's first all-electronic digital computer and one of its tasks was to produce projectile ballistic tables. This paper gives a five degree of freedom ballistic program for the hand-held SR-52 programmable calculator. You will find that the tables generated on the SR-52 are reasonably accurate.

A numerical integration method (some quadrature formula) is most frequently used to solve equation (1.1). This technique requires a reasonably large computer and it is long. The solution method described in

this paper is so simple that others must have discovered it before I did. In any case, I first discovered the iterative solution in 1963 while working as a consultant for the Fire Control Directorate at Frankford Arsenal. At this time, the Directorate was performing an error analysis on an AA weapon. While waiting for BRL ballistic tables to be generated we decided to generate our own to validate the error analysis model. However, the AA weapon error analysis was to be performed on BRL ballistic data. It was a pleasant surprise to find that the tables generated by our iterative method agreed with the BRL tables to within 5 meters in slant range for gun elevation angles between 20° and 80° and for slant ranges out to 2500 meters (maximum slant range for the error analysis). In the sequel you will observe that the accuracy of data depends on the iteration increment (Δt) and maximum slant range computed.

III. Solution of the Fundamental Ballistic Equation

In the cartesian X-Y plane, the fundamental ballistic equation becomes

$$(3.1) \quad \frac{d^2x}{dt^2} + H \frac{dx}{dt} = 0$$

$$(3.2) \quad \frac{d^2y}{dt^2} + H \frac{dy}{dt} + g = 0 .$$

As mentioned in the introduction, a solution to the above ballistic equations in closed form is impossible for the form of H given in equation (1.2). The parameters d , w and K_D of H are determined by the projectile. Frequently, K_D is given as a function of mach number as shown in Figure A.1. However, K_D may also be given as a function of remaining projectile speed. In either case a fourth degree polynomial fit is adequate.

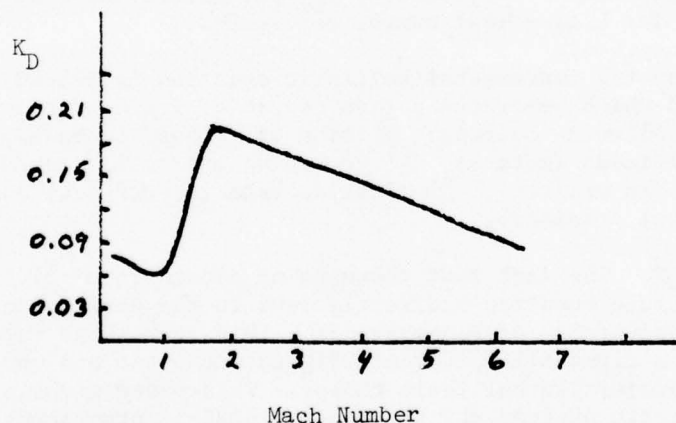


Figure A.1

Recall that ρ (air density) varies with altitude, air temperature and percentage of water vapor in the air, but ρ is mainly a function of altitude (a) for many ballistic projectiles. The program for the SR-52 uses the standard air density formula $\rho = \rho_0 e^{-hz}$ which is adequate for the range of altitudes usually considered. Observe that here ρ is a function of altitude only.

Equations (3.1) and (3.2) are solved under the assumption that H does not change during the time interval Δt which is a reasonable assumption when $0 \leq \Delta t < 0.01$. Under this assumption the ballistic equations are linear differential equations of order two with constant coefficients which have the following closed form solutions:

$$(3.3) \quad x = c_1 - c_1 e^{-Ht}$$

$$(3.4) \quad y = k_1 - k_1 e^{-Ht} - \frac{gt}{H}$$

for $0 \leq t < \Delta t$. The time derivative of each equation is

$$(3.5) \quad \frac{dx}{dt} = c_1 H e^{-Ht}$$

$$(3.6) \quad \frac{dy}{dt} = k_1 H e^{-Ht} - \frac{g}{H}$$

$$(3.7) \quad v = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}$$

for $0 \leq t < \Delta t$. Let $\phi(t)$ be defined by the equation

$$(3.8) \quad \tan \phi(t) = \frac{dy}{dx} / \frac{dx}{dt}$$

for $0 \leq t < \Delta t$. Then the arbitrary constants c_1 and k_1 are described by

$$(3.9) \quad c_1 = \frac{v \cos \phi}{H e^{-Ht}}$$

$$(3.10) \quad k_1 = \frac{v \sin \phi + g/H}{H e^{-Ht}}$$

for $0 \leq t < \Delta t$. The equations (3.3) through (3.10) are used to generate a ballistic table (see table attached) for a projectile for which d , w , K_D , V_0 (muzzle velocity) and ϕ_0 (gun firing angle) are known.

The incremental technique programmed (see Ballistic Program table) is the following. Let the muzzle of the gun be placed at the origin in the X-Y coordinate system and assume that the projectile is fired with a muzzle velocity v_0 at an angle ϕ_0 measured from the X-axis. The value of K_D at time t (initially $t = 0$) which is constant during the interval $0 \leq t < \Delta t$ is determined from the polynomial fit for K_D simply by substituting for v (initially $v = v_0$). Once K_D is determined, then H is determined using the parameters of the projectile, the present value of v and the value of ρ evaluated for the present altitude of the projectile. Equations (3.9) and (3.10) determine the arbitrary constants c_1 and k_1 in the interval $0 \leq t < \Delta t$ for the present value of v , ϕ and the value of H computed above. The position and velocity of the projectile at time $t = n\Delta t$ (initially $n = 1$), are derived from equations (3.3), (3.4), (3.5) and (3.6). For $n = 2$ the origin of the coordinate system is translated to the present position of the projectile just computed. Observe that equations (3.3) through (3.10) are solutions for the next time interval $\Delta t \leq t < 2\Delta t$ but with initial conditions determined by equations:

$$1) \quad t = 1\Delta t$$

$$2) \quad x = y = 0$$

$$3) \quad v = \sqrt{\left(\frac{dx}{dt}\bigg|_{t=\Delta t}\right)^2 + \left(\frac{dy}{dt}\bigg|_{t=\Delta t}\right)^2}$$

$$4) \quad \phi(\Delta t) = \tan^{-1}\left(\frac{dy}{dt}\bigg|_{t=\Delta t} / \frac{dx}{dt}\bigg|_{t=\Delta t}\right).$$

Now the process of the previous paragraph are repeated. After each iteration the projectile's position is found by summing x_i , y_i . That is:

$$\text{range} = \sum x_i, \quad \text{altitude} = \sum y_i, \quad \text{and slant range} = \sqrt{(\sum x_i)^2 + (\sum y_i)^2}.$$

IV. CONCLUSION. Although the ballistic solution described is almost trivial, the accuracy of the generated tabular data (see example of table on next page) is surprisingly good. The accuracy of the computed data can be shown to be a function of Δt for the maximum D computed. For $0 \leq \Delta t \leq 0.01$ you can expect the computed slant range data to be less than 10 meters for D out to 5000 meters. However, the most remarkable fact is that this method can be programmed on a hand-held SR-52 calculator.

TABLE

OEOPLIKOM

PHI = .872664676 RATIO = 1.177 VEL = 1175 M/SEC

WT = .550 KG DIA = .35 MET

DELTA = .010 SEC ALT = 0.0 MET

Time of flight	Speed of projectile	Drag coefficient	Range	Altitude	Range rate	Altitude rate	Slant Range	Angle of rise-fall
T	V	KD	SUMX	SUMY	XD	YD	D	PHI
.00	1175.0	.02170	0.0	0.0	0.0	0.0	0.0	.8727
.10	1142.4	.08297	74.5	88.7	734.8	874.8	115.9	.8721
.20	1111.2	.08424	147.0	175.0	715.2	850.4	228.5	.8716
.30	1081.2	.08549	217.6	258.9	696.4	827.1	338.1	.8710
.40	1052.4	.08672	286.3	340.4	678.3	804.8	444.4	.8704
.50	1024.7	.08792	353.2	419.8	660.0	783.0	548.6	.8698
.60	998.1	.08908	418.5	497.1	642.2	762.3	649.9	.8692
.70	972.5	.09021	482.1	572.3	624.2	742.4	749.3	.8685
.80	947.9	.09131	544.2	645.6	606.8	723.2	844.3	.8679
.90	924.3	.09238	604.7	717.0	589.0	704.3	937.0	.8672
1.00	901.6	.09342	663.8	786.5	570.8	687.1	1029.2	.8665
1.10	879.7	.09445	721.5	854.4	552.1	670.0	1119.3	.8658
1.20	858.6	.09546	777.8	920.6	532.9	653.5	1208.2	.8650
1.30	838.4	.09645	832.8	985.1	514.3	637.7	1295.0	.8643
1.40	818.8	.09744	886.7	1048.1	496.0	622.4	1379.9	.8635
1.50	800.0	.09843	939.7	1109.6	478.2	607.7	1463.2	.8627
1.60	781.8	.09941	990.7	1169.7	460.9	593.5	1545.2	.8619
1.70	764.2	.10040	1041.1	1228.3	444.0	579.7	1619.2	.8611
1.80	747.3	.10139	1090.4	1285.6	427.4	566.5	1685.7	.8603
1.90	730.9	.10239	1138.6	1341.6	411.2	553.6	1750.7	.8594
2.00	715.1	.10340	1185.8	1396.4	395.4	541.2	1811.9	.8585
2.10	699.8	.10442	1232.1	1449.9	380.2	529.2	1868.7	.8576
2.20	685.0	.10545	1277.4	1502.2	365.6	517.4	1921.0	.8567
2.30	670.6	.10649	1321.2	1553.4	351.7	506.3	1969.7	.8557
2.40	656.7	.10755	1365.3	1603.5	338.1	495.5	2016.0	.8547
2.50	643.2	.10863	1409.0	1652.5	324.7	484.8	2060.0	.8538
2.60	630.2	.10972	1449.9	1700.5	311.6	474.5	2103.7	.8527
2.70	617.5	.11082	1491.0	1747.4	298.7	464.4	2147.1	.8517
2.80	605.1	.11194	1531.2	1793.4	286.1	454.4	2189.2	.8507

BEST AVAILABLE COPY

TITLE BALLISTIC PROGRAM
PROGRAMMER

PAGE 1 OF Card #1
DATE 770203

SR-52
Coding Form

LOC	CODE	KEY	COMMENTS	LOC	CODE	KEY	COMMENTS	LOC	CODE	KEY	COMMENTS	LABELS
000	46	*LBL		01	1			42	STO	H		A
	15	E		40	*X ²			01	1			B
	43	RCL	A ₀₀	54	7			05	5			C
	00	0		95	=			20	*1/X			D
005	00	0		42	STO	K ₀		65	X			E
85	+			01	1			43	RCL	V		A
43	RCL	A ₀₁		02	2			01	1			B
00	0			43	RCL	SH		01	1			C
01	1			00	0			95	=			D
65	X			06	6			42	STO	V/H		E
43	RCL	V		42	STO	SH		01	1			REGISTERS
01	1			01	1			02	2			00
01	1			08	8			65	X			01
85	+			43	RCL	Σ _g		43	RCL	φ(radians)		02
43	RCL	V		01	1			01	1			03
01	1			00	0			06	6			04
01	1			42	STO	Σ _g		33	Cos			05
40	*X ²			01	1			95	=			06
65	X			09	9			42	STO	C _i		07
53	(51	SAR			00	0			08
43	RCL	A ₀₂		78	*5			07	7			09
00	0			65	X			43	RCL	V/H		10
02	2			43	RCL	ρ*		01	1			11
85	+			00	0			02	2			12
43	RCL	A ₀₃		05	5			65	X			13
00	0			65	X			43	RCL	φ(rad.)		14
03	3			43	RCL	d ² /w		01	1			15
65	X			01	1			06	6			16
43	RCL	V		04	4			32	Sin			17
01	1			65	X			95	=			18
01	1			43	RCL	V		85	+			19
85	+			01	1			43	RCL	9		FLAGS
43	RCL	A ₀₄		01	1			01	1			0
00	0			65	X			07	7			1
04	4			43	RCL	K ₀		55	÷			2
65	X			01	1			43	RCL	H		3
43	RCL	V		02	2							4
01	1			95	=							

TEXAS INSTRUMENTS
CORPORATION

SR-52 User Instructions

TITLE BALLISTIC PROGRAM

PAGE 1 of Card #1

$\leftarrow A =$	$\leftarrow B =$
------------------	------------------

STEP	PROCEDURE	ENTER	PHYSICS	DISPLAY
1	Enter constants: $K_D = P_g(V)$	a_{00} in S_{00}		EE 0.307815909 EE 00
		a_{01} in S_{01}		EE -7.298841830 EE-04
		a_{02} in S_{02}	$K_D = P_g(V)$	EE 1.005218520 EE-06
		a_{03} in S_{03}	for 35mm	EE -6.601229870 EE-10
		a_{04} in S_{04}		EE 1.63968606 EE-13
	P^* (Stan Air Density) $\frac{\text{Pibograms}}{\text{met}^3}$	P^* in S_{05}		EE 1.203368268 EE 00
		SH in S_{06}		EE -1.95433 EE-05
	$C_1 = \frac{V \cos \phi_i}{H e^{-H \Delta t}}$	C_1 in $S_{07}(R_{07})$		
	$K_1 = (V \sin \phi_i + \frac{g}{r}) / H e^{-H \Delta t}$	$D_1 = \frac{C_1}{K_1}$ in $S_{08}(R_{08})$		
	Σx (Range in meters)	Σx in $S_{09}(R_{09})$		0.0
	Σy (altitude in meters)	Σy in $S_{10}(R_{10})$		0.0
	V_i (remaining vel. $\frac{\text{met}}{\text{sec.}}$)	V_i in $S_{11}(R_{11})$	$V_0 =$	EE 1.175 EE 03
		$K_0 \rightarrow \frac{V}{H}$ in S_{12}		
	Δt (increment of time sec.)	Δt in S_{13}		
	d (diameter of projectile met)	$\frac{d^2}{w}$ in S_{14}		EE 2.227227 EE-03
	w (weight " " Pibograms)			
	$H = (P d^2 V K_D) / w$ ($\frac{\text{number}}{\text{sec.}}$)	H in $S_{15}(R_{15})$		
	ϕ_i (angle between vel. & ground)	ϕ_i in $S_{16}(R_{16})$	ϕ_0 (radians)	EE 8.72664626 EE-01
	g (gravity $\frac{\text{met}}{\text{sec.}^2}$)	g in S_{17}		EE 9.80665 EE 00
2	Initialize Program		E	
	To Find C_1		R_{07}	
	" " D_1		R_{08}	
	" " Σx		R_{09}	
	" " Σy		R_{10}	
	" " V_i		R_{11}	
	" " H		R_{15}	

© 1975 Texas Instruments Incorporated

TITLE BALLISTIC PROGRAM
PROGRAMMER

PAGE 2 OF Card #1
DATE 770203

SR-52
Coding Form



LOC	CODE	KEY	COMMENTS	LOC	CODE	KEY	COMMENTS	LOC	CODE	KEY	COMMENTS	LABELS
112	01	1		01	1			01	1			A
	05	5		85	+			00	0			B
	40	$\times x^2$		43	RCL	Σx		43	RCL	\bar{Y}		C
	95	=		00	0			01	1			D
	20	$\times \frac{1}{2}$		09	9			02	2			E
117	65	X		95	=			55	\div			A
	43	RCL	C ₁	42	STO	Σx		43	RCL	\bar{X}		B
	00	0		00	0			01	1			C
	07	7		09	9			01	1			D
	95	=		43	RCL	\bar{X}		95	=			E
122	42	STO	D ₁	01	1			22	INV			REGISTERS
	00	0		01	1			34	Tan			09
	08	8		55	\div			42	STO	$\phi(\text{rad.})$		01
	43	RCL	H	43	RCL	D ₁		01	1			02
	01	1		00	0			06	6			03
127	05	5		08	8			81	NLT			04
	42	STO	H	75	-			46	*LBL			05
	01	1		43	RCL	Δx		78	*5			06
	08	8		01	1			01	1			07
	43	RCL	Δx	03	3			22	INV			08
132	01	1		65	X			23	Rn X			09
	03	3		43	RCL	9		45	y^x			10
	42	STO		01	1			53	(11
	01	1		07	7			43	RCL			12
	09	9		55	\div			01	1			13
137	51	SBR		43	RCL	H		08	8			14
	78	*5		01	1			65	X			15
	94	+/-		05	5			43	RCL			16
	85	+		95	=			01	1			17
	01	1		42	STO	\bar{Y}		09	9			18
142	95	=		01	1			94	+/-			19
	65	X		02	2			54)			FLAGS
	43	RCL	C ₁	85	+			95	=	e^{x^2}		0
	00	0		43	RCL	Σy		56	*atan			1
	07	7		01	1							2
147	95	=		00	0							3
	42	STO	\bar{X}	95	=							4
	01	1		42	STO	Σy		TEXAS INSTRUMENTS INCORPORATED				

此

PAGE 2 OF Card #1

[illegible]

STEP	PROCEDURE	ENTER	PRESS	DISPLAY
	To Find ϕ_i		R_{16}	
3	Put in Card #2		B'	D
	To Find D (slant Range)		C'	
	To Find V_{i+1}		R_{11}	V_{i+1}
	" " \bar{x}		R_{19}	
	" " \bar{s}		R_{18}	

TITLE BALLISTIC PROGRAM
PROGRAMMER

PAGE 1 OF Card #2
DATE 770203

SR-52
Coding Form

LOC	CODE	KEY	COMMENTS	LOC	CODE	KEY	COMMENTS	LOC	CODE	KEY	COMMENTS	LABELS
000	✓ 46	*LBL			65	X			22	INV		A
	17	B'			43	RCL	C.		23	ln x		B
	43	RCL	Σx	040	✓ 00	0			45	y ^x		C
	00	0			07	7			53	(D
	09	9			95	=			✓ 43	RCL	\dot{y}	E
005	✓ 42	STO			42	STO	\dot{x}		01	1		A
	01	1			01	1			08	8		B
	08	8		045	✓ 09	9			65	X		C
	43	RCL	Σy		43	RCL	\dot{x}		43	RCL	\dot{x}	D
	01	1			01	1			✓ 01	1		E
010	✓ 00	0			09	9			09	9		REGISTERS
	42	STO			55	÷			94	+/-		09
	01	1		050	✓ 43	RCL	D.		54)		10
	09	9			00	0			95	=		11
	51	SBR	$\sqrt{(Sx)^2 + (Sy)^2}$		08	8			✓ 56	*atan		12
015	✓ 87	*1			75	-			46	*LBL		13
	95	=			43	RCL	9		87	*1		14
	81	HLT		055	✓ 01	1			53	(15
	46	*LBL			07	7			43	RCL	$\dot{y}, \Sigma y$	16
	18	C'			55	÷			✓ 01	1		17
020	✓ 43	RCL	H		43	RCL	H		08	8		18
	01	1			01	1			40	*X ²		19
	05	5		060	✓ 05	5			85	+		20
	42	STO			95	=	\dot{y}		43	RCL	$\dot{x}, \Sigma x$	21
	01	1			42	STO			✓ 01	1		22
025	✓ 08	8			01	1			09	9		23
	42	RCL	Δt		08	8			40	*X ²		24
	01	1		065	✓ 51	SBR	$\sqrt{X_1^2 + X_2^2}$		54)		25
	03	3			87	*1			30	*Y		26
	42	STO			95	=			✓ 95	=		27
030	✓ 01	1			42	STO	Y_{i+1}		56	*atan		28
	09	9			01	1						29
	51	SBR		070	✓ 08	1						FLAGS
	78	*5			95	=						30
	65	X			81	HLT						31
035	✓ 43	RCL	H		46	*LBL						32
	01	1			78	*5						33
	05	5		075	✓ 01	1						34

TEXAS INSTRUMENTS
INCORPORATED

15

PAGE 1 OF Card #2

[illegible]

APPROXIMATE DECOMPOSITIONS OF SPARSE MATRICES

Henk A. van der Vorst
Academic Computer Center Utrecht
Budapestlaan 6
De Uithof - Utrecht
Netherlands

ABSTRACT. The discretisation of self-adjoint elliptic partial differential equations often leads to large linear systems with a sparse matrix. An attractive way of solving such systems arises when an approximate decomposition of the matrix is used in the application of the conjugate gradients method. Some of these approximate decompositions will be treated in more detail.

1. INTRODUCTION. We consider large linear systems

$$(1) \quad Ax = b$$

where A is a sparse symmetric matrix. More explicitly, the matrix A is assumed to arise from 5-point finite difference discretisation of second order self-adjoint partial differential equations. However, most of the ideas can be extended to other types of sparse matrices. Both direct and iterative methods, that take advantage of the special sparse structure of A have been studied for the solution of (1).

For direct methods we mention papers of George [1,2] and Buzbee et al [3,4]. Iterative methods have been studied by Varga [5], Young [6], Concus & Golub [7,8], O'leary [9], Axelsson [10,11] and many others. In Bunch & Rose [12] and in Barker [13] many recent techniques and results for direct methods as well as iterative methods are covered and further references may be found there. The results presented in this paper are based on ideas outlined by Meijerink & van der Vorst [14]. They propose iterative methods that exploit certain approximate decompositions of the matrix A . These decompositions can be represented in a sparse way and are in general easy to construct and cheap to compute. If the matrix A is positive definite, then the conjugate gradients method may be used in connection with these approximations, which in general results in very fast iterative methods.

In this paper a number of approximate decompositions for some special structured matrices are demonstrated. From these applications it follows clearly how to construct decompositions for other types of matrices. Also the convergency-properties of the different methods are treated in some way and computational aspects are discussed briefly.

2. THE ITERATIVE PROCEDURE. Unless otherwise stated, it will be assumed throughout this paper that the matrix $A=(a_{ij})$ is a symmetric M-matrix. A matrix A is an M-matrix if $a_{ij} < 0$ for $i \neq j$, A^{-1} exists and $A^{-1} \geq 0$ (Varga [1, 15]).

The linear system (1) might be solved by a decomposition method for the matrix A . In general the decomposition factors are far less sparse than A , and therefore it appears to be attractive to look for approximate decompositions where the factors are sufficiently sparse.

The existence of such sparse factorizations is guaranteed by a theorem given in [14]. That theorem states that a sparsity pattern for the factor L (in $A=LL^T+R$) can be chosen in advance. Since $K=LL^T$ is a positive definite symmetric matrix, this approximate decomposition can be used in combination with the conjugate gradients method.

The resulting iterative scheme can be written as:

x_0 is an arbitrary initial approximation to x

$$r_0 = b - Ax_0, \quad p_0 = K^{-1}r_0$$

$$\alpha_i = \frac{(r_i, K^{-1}r_i)}{(p_i, Ap_i)}$$

(2)

$$x_{i+1} = x_i + \alpha_i p_i$$

$$r_{i+1} = r_i - \alpha_i Ap_i$$

$$\beta_i = \frac{(r_{i+1}, K^{-1}r_{i+1})}{(r_i, K^{-1}r_i)}$$

$$p_{i+1} = K^{-1}r_{i+1} + \beta_i p_i$$

$i=0, 1, 2, \dots$

It should be noted that K^{-1} will not be determined explicitly, vectors $y=K^{-1}z$ will be determined by solving $LL^Ty=z$ in two steps. For details see [14].

3. TWO-DIMENSIONAL PROBLEMS. In this section we assume the matrix to arise from 5 point finite difference discretisation of

$$(3) \quad -(\alpha u_x)_x - (\alpha u_y)_y + \gamma u = \delta, \quad ,$$

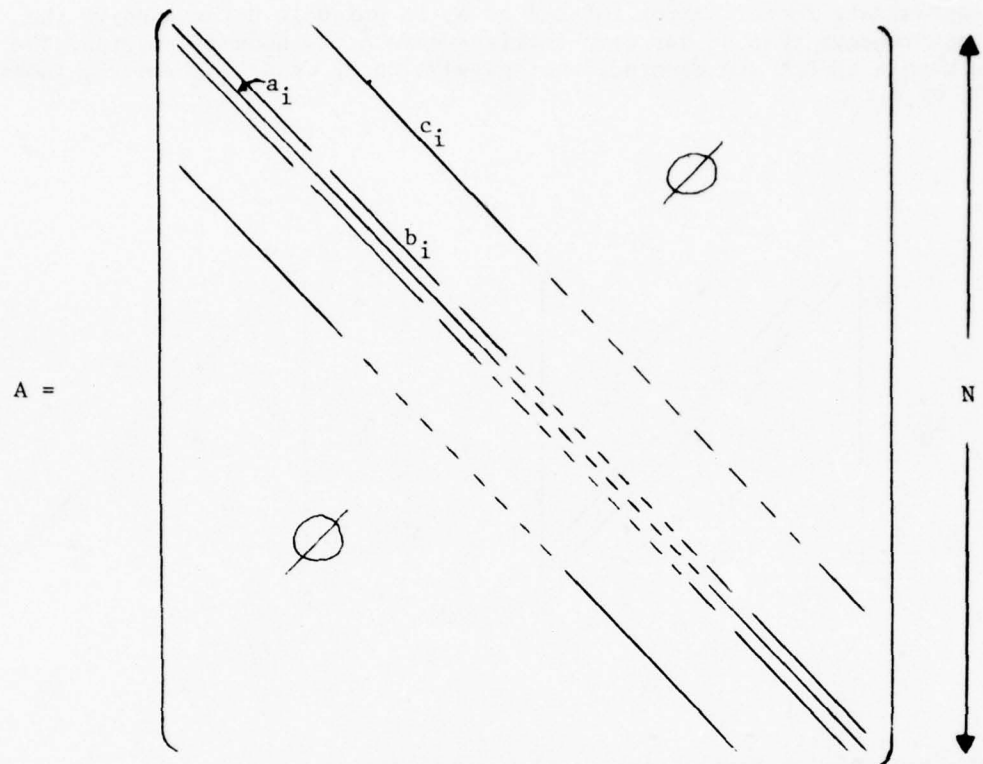
$$\alpha(x,y) > 0, \quad \gamma(x,y) \geq 0 \quad \text{over } R, \quad ,$$

and suitable boundary conditions along ∂R .

For R we take a rectangular region in the (x,y) -plane.

This leads to matrices with the desired properties (see section 1).

The form of this matrix is:



The elements of the uppertriangular part of A are denoted by a_i , b_i and c_i , where i is counted rowwise. The order of the matrix is given by N .

3.1. $K=DIAG(A)$. The simplest approximation K of A arises when we choose all off-diagonal elements of L (in $K=LL^T$) to be zero. Obviously then K is a diagonal matrix whose diagonal elements are equal to a_i (the diagonal elements of A). The hybrid conjugate gradient method, that results from this choice, is equivalent with the conjugate gradients algorithm applied directly on the scaled matrix A , where the diagonal elements of A are used as scalings factors. This scaling is in some sense optimal for the conjugate gradients method since it minimises among all scalings the condition number of A [16].

3.2. $ICCG(0)$. A more efficient method results if we choose the matrix K in such a way, that its factors are as sparse as A itself. It is convenient in this case to write the approximation as $K=L_0D_0L_0^T$, where L_0^T is an upper triangular matrix and D_0 is a diagonal matrix. In this way square roots are avoided, but the main advantage is the property that this factorization can be represented by only one diagonal.

We have

$$(4) \quad A = L_0 D_0 L_0^T + R_0 \quad (K = L_0 D_0 L_0^T)$$

If we choose D_0 to be the inverse of the diagonal of L_0^T , then this approximate factorization $L_0 D_0 L_0^T$ of A , is uniquely determined by the requirements that R_0 has zero entries where A has nonzero entries. The elements of L_0^T are denoted, analogously to A , by \tilde{a}_i , \tilde{b}_i and \tilde{c}_i , those of D_0 by \tilde{d}_i :

$$L_0^T = \begin{bmatrix} & & \\ & \tilde{a}_i & \\ & \tilde{b}_i & \\ & & \tilde{c}_i & \\ & & & & \end{bmatrix} \quad D_0 = \begin{bmatrix} & & \\ & & \\ & & \tilde{d}_i & \\ & & & & \end{bmatrix}$$

The following relations are easily verified:

$$(5) \quad \begin{aligned} \tilde{a}_i &= \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m} \\ \tilde{b}_i &= b_i, \quad \tilde{c}_i = c_i \end{aligned}$$

for $i=1, \dots, N$, where m is the half band-width of A . Not defined elements should be replaced by zero's.

From the relations (5) it follows that only the elements \tilde{d}_i need to be stored.

The resulting hybrid conjugate gradients method is called ICCG(0) [14]. The computational efforts for this iterative method are roughly proportional to $16N$ multiplications per iteration.

3.3. ICCG(1). The next approximation is defined by

$$(6) \quad A = L_1 D_1 L_1^T + R_1 \quad (K_1 = L_1 D_1 L_1^T)$$

and R_1 is required to have zero's where $A - R_0$ (see 3.2) has nonzero's, D_1 is a diagonal matrix whose elements are equal to the inverses of the diagonal-elements of L_1^T . From this it follows that L_1^T has one extra non-zero

diagonal as compared to L_0^T :

$$L_1^T = \begin{bmatrix} & & & \\ & \tilde{a}_i & \tilde{b}_i & \\ & & \tilde{f}_i & \tilde{c}_i \\ & & & \ddots \end{bmatrix} \quad D_1 = \begin{bmatrix} & & & \\ & & & \\ & & d_i & \\ & & & \ddots \end{bmatrix}$$

The elements are defined by:

$$\begin{aligned} \tilde{a}_i &= \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{f}_{i-m+1}^2 \tilde{d}_{i-m+1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m} \\ \tilde{b}_i &= b_i - \tilde{c}_{i-m+1} \tilde{f}_{i-m+1} \tilde{d}_{i-m+1} \\ \tilde{f}_i &= -\tilde{c}_{i-1} \tilde{b}_{i-1} \tilde{d}_{i-1} \\ \tilde{c}_i &= c_i \end{aligned} \quad (7)$$

for $i=1,2,\dots,N$, where again m is the half band-width of A , and not defined elements should be replaced by zero's. Storage requires now 3 extra vectors of length N , and the computational efforts of the resulting hybrid conjugate gradients method are proportional to $18N$ multiplications per iteration.

3.4. ICCG(3). If we proceed in this manner, then the next approximation is characterized by:

$$(8) \quad A = L_3 D_3 L_3^T + R_3 \quad (K=L_3 D_3 L_3^T)$$

This approximation follows from the requirements that R_3 has zero's where $A-R_0-R_1$ has non zero elements, and from the choice of D_3 to be a diagonal matrix equal to the inverse of the diagonal of L_3^T .

$$L_3^T = \left[\begin{array}{ccc} & & \\ & \tilde{a}_i & \\ & \tilde{b}_i & \\ & \tilde{g}_i & \\ & & \tilde{e}_i, \tilde{f}_i, \tilde{c}_i \\ & & & \end{array} \right]$$
$$\tilde{d}_i^{-1} = \tilde{a}_i = a_i - b_{i-1}^2 \tilde{d}_{i-1} - \tilde{g}_{i-2}^2 \tilde{d}_{i-2} - \tilde{e}_{i-m+2}^2 \tilde{d}_{i-m+2} - \tilde{f}_{i-m+1}^2 \tilde{d}_{i-m+1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m}$$

$$(9) \quad \tilde{b}_i = b_i - \tilde{g}_{i-1} \tilde{b}_{i-1} \tilde{d}_{i-1} - \tilde{c}_{i-m+1} \tilde{f}_{i-m+1} \tilde{d}_{i-m+1} - \tilde{f}_{i-m+2} \tilde{e}_{i-m+2} \tilde{d}_{i-m+2}$$

$$\tilde{g}_i = -\tilde{c}_{i-m+2}\tilde{e}_{i-m+2}\tilde{d}_{i-m+2}$$

$$\tilde{e}_i = - (\tilde{c}_{i-2} \tilde{g}_{i-2} \tilde{d}_{i-2} + \tilde{f}_{i-1} \tilde{b}_{i-1} \tilde{d}_{i-1})$$

$$\tilde{f}_i = -\tilde{c}_{i-1}\tilde{b}_{i-1}\tilde{d}_{i-1}$$

$$\tilde{c}_j = c_j$$

An extra storage of 5 vectors of length N is required to store L_3^T and the computational efforts of the resulting hybrid conjugate gradients method are proportional to $22N$ multiplications per iteration.

3.5. HOW FAR? The process of constructing better approximations can be continued of course until we have a complete factorization. It will be evident that there is, depending on the problem, some optimum choice somewhere between the most sparse approximation and the complete factorization. The factorization next to the one in section 3.4, would have been

$$(10) \quad A = L_7 D_7 L_7^T + R_7$$

where L_7^T has 7 extra non-zero diagonals as compared to the equivalent

part of A. The representation would take 9 extra vectors of length N and the computational costs would be proportional to 30N multiplications per iteration.

Experiments show that, compared to "pure" conjugate gradients (i.e. K=I) for the solution of $Ax=b$, the biggest improvement is made by ICCG(0), and in most cases the "optimum", with respect to computational costs, is reached by ICCG(3).

4. EIGENVALUES OF $K^{-1}A$. For a better understanding of these methods it is instructive to see how the eigenvalues of $K^{-1}A$ behave. From straightforward computations it follows that the iterative processes based on approximate decompositions, in combination with the conjugate gradients algorithm for the solution of $Ax=b$, are equivalent with conjugate gradients applied on the linear system

$$(11) \quad L^{-1}D^{-\frac{1}{2}}AD^{-\frac{1}{2}}L^{-1}y = L^{-1}D^{-\frac{1}{2}}b, \quad x = D^{\frac{1}{2}}L^Ty,$$

where $K=LDL^T$.

For conjugate gradients we have that

$$(12) \quad \|x_i - x\|_A \leq \alpha^i \|x_0 - x\|_A$$

where
$$\|x_i - x\|_A^2 \equiv (x_i - x, A(x_i - x))$$

$$(13) \quad \alpha = \frac{\sqrt{c} - 1}{\sqrt{c} + 1} \quad c = \frac{\lambda_{\max}}{\lambda_{\min}},$$

and λ_{\max} , λ_{\min} are the largest and smallest eigenvalues of $K^{-1}A$. However, this upperbound α for the rate of convergence appears to be far to pessimistic in general, since the influence of components in the starting vector x_0 , in directions of eigenvectors, damps out soon.

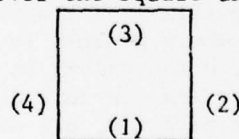
On the other hand, it is well-known, that the conjugate gradients algorithm takes full advantage of relative clustering of the eigenvalues [17]. This is the reason that the new algorithms are so successful, because premultiplication with the approximate inverse K^{-1} of the matrix A, forces most of the eigenvalues of $K^{-1}A$ towards 1.

This will be demonstrated by an example. Consider the linear system arising from discretisation of $\Delta u=0$ over the square unit region.

The boundary conditions are:

$\frac{\partial u}{\partial n} = 0$ along (2), (3) and

(4), and u is a given function along (1).



A grid is chosen with meshwidths $\frac{1}{31}$ in each direction, which results in a linear system with 992 unknowns ($N=992, m=31$).

In order to demonstrate the effects on the eigenvalues, depending on the choice of approximation, we listed in table 1 the five smallest and the five largest eigenvalues of A and of $K^{-1}A$ for ICCG(0), ICCG(1) and ICCG(3).

nr eigenvalue	A	ICCG(0)	ICCG(1)	ICCG(3)
1	0.0025	0.0045	0.0119	0.0252
2	0.0121	0.0219	0.0589	0.1241
3	0.0223	0.0391	0.0983	0.1900
4	0.0319	0.0561	0.1428	0.2740
5	0.0409	0.0712	0.1840	0.3614
.	.			
.	.			
.	.			
988	7.904	1.2106	1.1815	1.1583
989	7.922	1.2108	1.1861	1.1595
990	7.951	1.2119	1.1885	1.1666
991	7.952	1.2120	1.1888	1.1747
992	7.980	1.2324	1.1934	1.1794

Table 1.

In table 2 the percentage of eigenvalues that are in the interval [0.85, 1.15] are given.

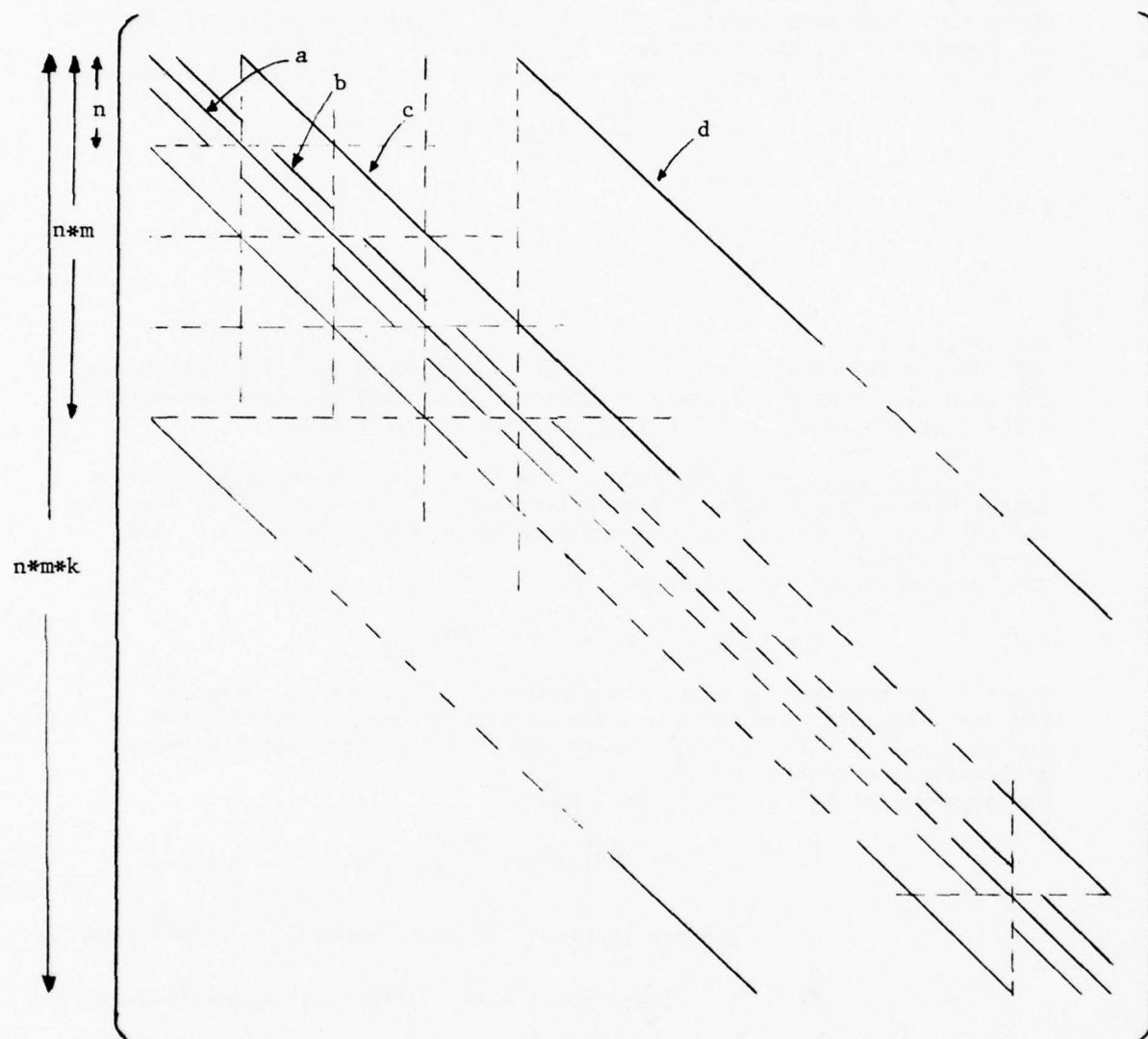
matrix	percentage in [0.85, 1.15]
A	~ 40 %
ICCG(0)-matrix	74 %
ICCG(1)-matrix	92 %
ICCG(3)-matrix	97 %

Table 2.

5. 3-DIMENSIONAL PROBLEMS. In this section, two approximate decompositions will be given for symmetric linear systems that arise from discretisation of 3-dimensional partial differential equations

$$(14) \quad -(\alpha u'_x)'_x - (\alpha u'_y)'_y - (\alpha u'_z)'_z + \gamma u = \delta$$

over a rectangular region, and with suitable boundary conditions. The N-th order matrix A of the resulting linear system $Ax=b$ has the following form, where $N=n \times m \times k$ (n is the number of grid points in the x-direction, m the number in the y-direction and k the number in the z-direction). The non-zero diagonals of the upper triangular part, of the symmetric matrix A, are denoted by a, b, c and d respectively, subscripts are counted rowwise.



Form of the matrix A for three-dimensional problems.

5.1. ICCG(0) - 3 DIMENSIONAL. Our first approximate factorization is defined by

$$(15) \quad A = L_0 D_0 L_0^T + R_0 \quad (K=L_0 D_0 L_0^T)$$

where L_0 and L_0^T are as sparse as A , and R_0 has zero elements on places where A has non zero elements. The elements of the diagonal matrix D_0 are denoted by Δ_i , the elements of L_0^T by \tilde{a}_i , \tilde{b}_i , \tilde{c}_i and \tilde{d}_i . The values of these elements can be computed from the following relations:

$$(16) \quad \begin{aligned} \Delta_i^{-1} &= \tilde{a}_i = a_i - \tilde{b}_{i-1}^2 \Delta_{i-1} - \tilde{c}_{i-n}^2 \Delta_{i-n} - \tilde{d}_{i-n*m}^2 \Delta_{i-n*m} \\ \tilde{b}_i &= b_i \\ \tilde{c}_i &= c_i \\ \tilde{d}_i &= d_i \end{aligned}$$

for $i=1,2,\dots,N$.

Only one vector of the length N is required to store this factorization. The resulting hybrid conjugate gradients method requires computational efforts proportional to $20N$ multiplications per iteration.

5.2. ICCG(3) - 3 DIMENSIONAL. The next (and better) approximation arises when an approximate decomposition $L_3 D_3 L_3^T$ is constructed, where L_3 and L_3^T have non-zero elements on places where $A-R_0$ (section 5.1) has non-zero elements.

This decomposition is defined by

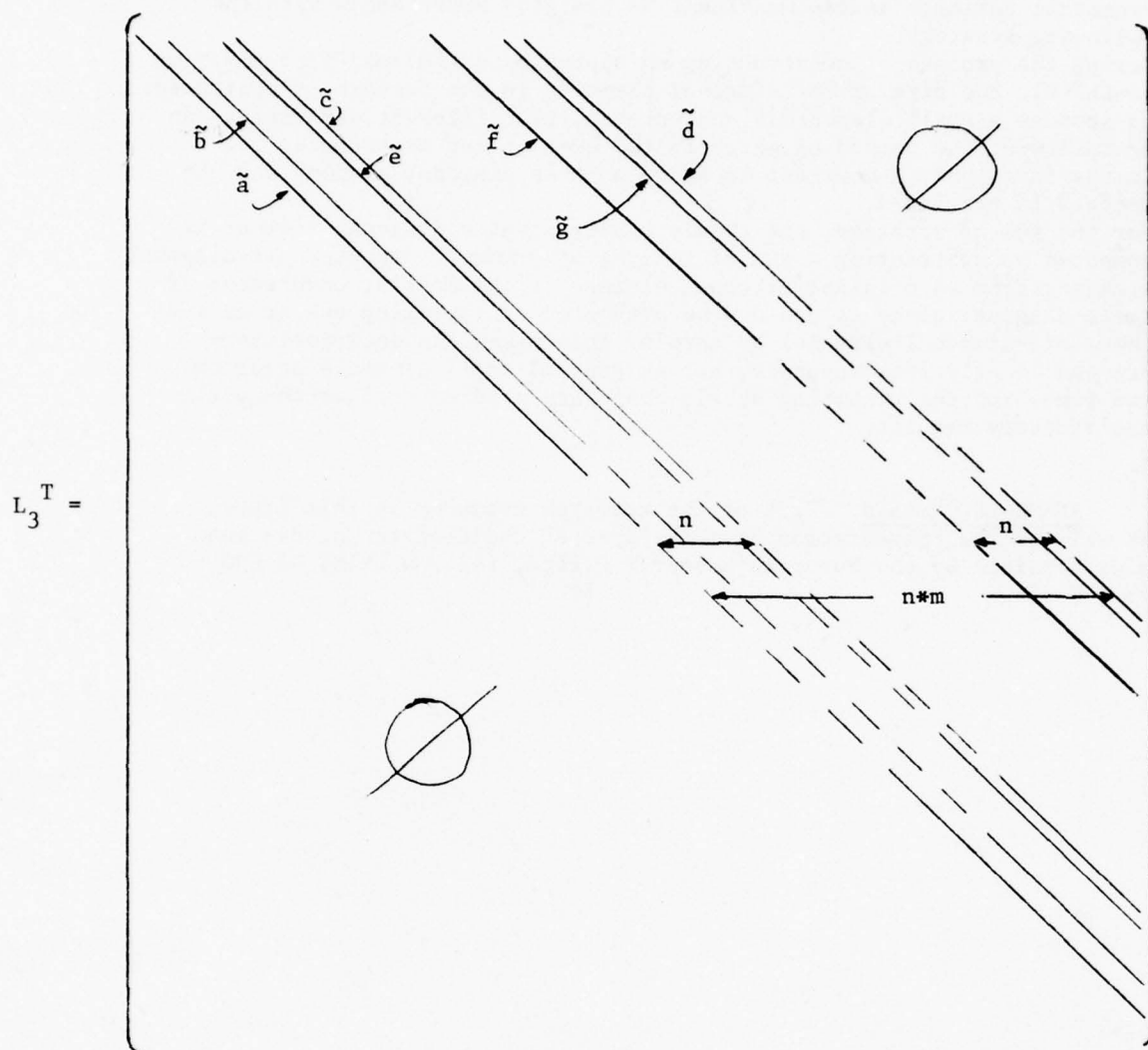
$$(17) \quad A = L_3 D_3 L_3^T + R_3 \quad (K=L_3 D_3 L_3^T)$$

Where R_3 is required to have zero elements on places where $A-R_0$ has non-zero elements, and D_3 is a diagonal matrix equal to the inverse of the diagonal of L_3^T . It then appears that L_3^T has three extra non-zero diagonals as compared to L_0^T .

The elements of L_3^T and D_3 follow from:

$$(18) \quad \begin{aligned} \tilde{a}_i &= \Delta_i^{-1} = a_i - \tilde{b}_{i-1}^2 \Delta_{i-1} - \tilde{e}_{i-n+1}^2 \Delta_{i-n+1} - \tilde{c}_{i-n}^2 \Delta_{i-n} \\ &\quad - \tilde{f}_{i-n*m+n}^2 \Delta_{i-n*m+n} - \tilde{g}_{i-n*m+1}^2 \Delta_{i-n*m+1} - \tilde{d}_{i-n*m}^2 \Delta_{i-n*m} \\ \tilde{b}_i &= b_i - \tilde{c}_{i-n+1} \Delta_{i-n+1} \tilde{e}_{i-n+1} - \tilde{d}_{i-n*m+1} \Delta_{i-n*m+1} \tilde{g}_{i-n*m+1} \\ \tilde{e}_i &= -\tilde{c}_{i-1} \Delta_{i-1} \tilde{b}_{i-1} - \tilde{g}_{i-n*m+n} \Delta_{i-n*m+n} \tilde{f}_{i-n*m+n} \\ \tilde{c}_i &= c_i - \tilde{d}_{i-n*m+n} \Delta_{i-n*m+n} \tilde{f}_{i-n*m+n} \\ \tilde{f}_i &= -\tilde{g}_{i-n+1} \Delta_{i-n+1} \tilde{e}_{i-n+1} - \tilde{d}_{i-n} \Delta_{i-n} \tilde{c}_{i-n} \\ \tilde{g}_i &= -\tilde{d}_{i-1} \Delta_{i-1} \tilde{b}_{i-1} \\ \tilde{d}_i &= d_i \end{aligned}$$

where $i=1,2,\dots,N$, and L_3^T has the following form:



Six vectors of length N are required to store the non-zero diagonals of L_3^T . The resulting hybrid conjugate gradients method requires computational efforts proportional to $26N$ multiplications per iteration. It has been observed that in general ICCG(3) - 3D gives a remarkable improvement over ICCG(0) - 3D, and it should be mentioned here that leaving away one or more of the diagonals \tilde{e} , \tilde{f} and \tilde{g} , in general results in an iterative method which is only little faster than ICCG(0) - 3D.

6. ADDITIONAL REMARKS. From the examples given in the previous section, it is easy to see how to handle matrices with different structures. Especially, the variants where the factorizations are as sparse as the matrix in question, are extremely simple to compute.

All those approximations cause no trouble as long as the original matrix is an M-matrix. It should be stressed here that positive definiteness is not enough. However, some statements can be made about symmetric definite sparse matrices. We had good experiences with the following strategy.

During the process of constructing an approximate Choleski-factorization ($A=LL^T+R$), the size of the diagonal elements in the factors is monitored. As soon as a small element is encountered, two different strategies can be followed, the second of which is the best in our experience.

In the first one, a constant is added to this diagonal element and the process is continued.

For the second strategy, one should realize that a diagonal element is computed by subtracting a sum of squares of already computed off-diagonal elements from an original diagonal element of the matrix. Occurrence of small diagonal elements can now be prevented by replacing one or more of these off-diagonal elements by zero's. This makes the decomposition-process locally less accurate, but in general small elements occur only few times and the resulting hybrid conjugate gradients algorithm yields satisfactory results.

ACKNOWLEDGEMENTS. Part of the research reported in this paper, as well as the presentation of this paper at the Conference, has been made possible by the European Research Office, through GRANT DA ERO - 75 - G 084.

REFERENCES

1. Alan George, "Nested dissection of a regular finite element mesh", SIAM J. Numer. Anal., 10 (1973), pp 345-363.
2. Alan George, "Solution of linear system equations: direct methods for finite element problems",
in: V.A.Barker, ed., "Sparse Matrix Techniques", Springer Verlag, Berlin, 1977.
3. B.L.Buzbee, G.H.Golub and C.W.Nielson, "On direct methods for solving Poisson's equations, SIAM J. Numer. Anal., 7 (1970), pp 627-656.
4. B.L.Buzbee, F.W.Dorr, J.A.George and G.H.Golub, "The direct solution of the discrete Poisson equation on irregular regions", SIAM J. Numer. Anal., 8 (1971), pp 722-736.
5. R.S.Varga, "Matrix Iterative Analysis", Prentice-Hall, Englewood Cliffs, N.J., 1962.
6. D.M.Young, "Iterative solution of large linear systems", Academic Press, New York, 1971.
7. P.Concus and G.H.Golub, "Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations", SIAM J. Numer. Anal., 10 (1973), pp 1103-1120.
8. P.Concus, G.H.Golub and D.P.O'leary, "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations",
in: J.R.Bunch and D.J.Rose, ed., "Sparse Matrix Computations", Academic Press, New York, 1976.
9. D.P.O'leary, "Hybrid conjugate gradient algorithms", Ph.D.Thesis, Comp. Science Dept., Stanford Univ., 1975.
10. O.Axelsson, "A generalized SSOR method", BIT 13 (1972), pp 443-467.
11. O.Axelsson, "A class of iterative methods for finite element equations", Dept. of Comp. Science, Göteborg Univ., 1975.
12. J.R.Bunch and D.J.Rose, ed., "Sparse Matrix Computations", Academic Press, New York, 1976.
13. V.A.Barker, ed., "Sparse Matrix Techniques", Springer Verlag, Berlin, 1977.
14. J.A.Meijerink and H.A.van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix", Math. of Comp., 31 (1977), pp 148-162.
15. R.S.Varga, "M-matrix theory and recent results in numerical linear Algebra",
in: J.R.Bunch and D.J.Rose, "Sparse Matrix Computations", Academic Press, New York, 1976.
16. A.van der Sluis, "Condition numbers and equilibration of matrices", Numer. Math. 14 (1969), pp 14-23.
17. O.Axelsson, "Solution of linear systems of equations: iterative methods",
in: V.A.Barker, ed. "Sparse Matrix Techniques", Springer Verlag, Berlin, 1977.

COMMENTS ON THE SOLUTION OF COUPLED STIFF DIFFERENTIAL EQUATIONS

M. D. Kregel

J. M. Heimerl

U.S. Army Ballistic Research Laboratory
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. The K-method of integrating sets of ordinary differential equations is outlined. This algorithm employs a variable step size, third order predictor-corrector method. It is written to reduce truncation error and to maximize stability consistent with reasonable execution time. The K-method has been used to integrate equations with discontinuous driving functions and this algorithm conserves chemical balance within the machine round-off error. It has been applied to solve kinetics problems in aeronomy and examples are taken from that field.

INTRODUCTION. In the late 1960's the aeronomy branch at the BRL needed the solutions to sets of stiff ordinary differential equations (ODES) that describe the positive and negative ion chemistry in the earth's D-region (~ 60-85 km). Adequate mathematical techniques for handling stiff ODES were unknown to us at that time. Kregel, a physicist, approached this problem empirically and developed a stiff ODE integrator. This report sketches the K-method of integration and provides a few examples of the uses of this integrator. Our aim is two-fold: to interest mathematicians so that this method may be placed on a firmer mathematical foundation and to inform potential users so that they may apply this algorithm to their particular problem.

TWO BASIC PROBLEMS. Before we discuss the sketch and the examples let us briefly review two problems that are basic to any numerical solution to ODES, stiff or not. These problems are truncation error and stability. According to Dahlquist and Björck,¹..."(truncation errors) are committed when a limiting process is broken off before one has come to the limiting value." Truncation errors result from mathematical approximations. For example they arise when a finite series approximates an infinite series or when a linear function approximates a non-linear one.

Stability, or its better known opposite, instability, is associated with the idea of feedback.² As the name implies, part of a program or code has a loop in which the numbers produced at the output of one cycle are used as the input for the next cycle. The errors associated with these numbers may then be amplified in such a way as to destroy the solution.

¹Numerical Methods, by G. Dahlquist and A. Björck, Trans. by N. Anderson, 1974, Prentice Hall, Inc., Englewood Cliffs, NJ, p. 22.

²See for example Numerical Methods for Scientists and Engineers, by R. W. Hamming, 2nd Edition, 1973, McGrawHill, Inc. p. 5.

The purpose in recalling these nemeses is to justify the effort that has gone into the K-method algorithm to reduce truncation error and to maximize stability consistent with a reasonable execution time.

SKETCH OF THE K-METHOD. Figure 1 schematically shows the main functional steps in the K-method algorithm.³ A third-order predictor-corrector method is employed. As soon as the initial corrector is formed, the diagonal of the Jacobian (i.e. $\partial y_i' / \partial y_i$) is examined to determine those dependent variables that are stiff and those that are not (not shown in Fig. 1). This is done in order to select the method with the least computational overhead for updating each of the initial predictor values.

At this stage of the algorithm neither the predictor values nor corrector values are presumed acceptable and an error vector is generated from their difference. This vector, in conjunction with the Jacobian including, now, the off-diagonal elements, is used to modify the predictor values. The modification to the predicted values, which involve a matrix inversion, are first attempted iteratively using a Gauss-Seidel method. During the iteration, checks are made on the estimated computational overhead burden. Should the iteration method prove too tedious, the remaining "non-converged" correction elements are solved by direct matrix inversion. The corrector is recomputed and another error vector is generated.

Truncation error is then checked by comparing the fourth derivative of each dependent variable against a predetermined relative error tolerance. If any one of these variables fails this test, the truncation error is judged "poor", the step size, h , is reduced by a factor of two and this cycle of the computation is begun anew. This test is especially useful, as we shall see later, in the case of discontinuous driving functions.

When all the dependent variables have passed the truncation test, (i.e., are "OK" in Fig. 1) a check is made on the predictor-corrector agreement. Should all elements of the predictor-corrector difference vector be less than a predetermined minimum error (i.e. are "OK"), the corrector values are accepted as the solutions at this time step and the step size is adjusted for the next cycle. If any element of this difference vector is greater than a predetermined maximum error tolerance the agreement is judged "poor," the step size is reduced by a factor of two and this cycle begun anew. For the intermediate case, in which all elements of this difference vector are less than the maximum error test, and in which at least one is greater than the minimum error test, the difference vector is judged "so-so". When this is the case the predicted

³An earlier version has been reported. See "Description and Comparison of the K-Method for Performing Numerical Integration of Stiff Ordinary Differential Equations," by M. D. Kregel and E. L. Lortie, BRL Report No. 1733, July 1974, AD #A003855.

values are again modified and the process begun anew until an acceptable solution is obtained. It is important to note that the K-method conserves both charge, if any, and chemical balance to within the round-off error of the machine since the corrector formulation itself is conservative.

The methodology outlined in Figure 1 and above shows that a solution over one time step is considered valid if the corrected values map into the predicted values within a specified error tolerance. Since small differences in the predicted values are magnified by a factor of the order of the stiffness, we have attempted to formulate a predictor-corrector scheme which has minimum error and maximum stability. How this has been empirically achieved is outlined below.

To be solved are vector equations which may be cast in the form of

$$Y' = F - RY, \quad (1)$$

where Y is the dependent vector, and a function of the independent variable X , F the formation vector [$= F(Y,X)$] and R the removal vector [$= R(Y,X)$]. The predictor is chosen to be quadratic in form; i.e.,

$$Y_1^P = Y_0 + A(X_1 - X_0) + B(X_1 - X_0)^2, \quad (2)$$

where $(X - X_0)$ is the local step size and where the subscript zero denotes the current location. A and B are to be determined. To do this we require two independent equations. One is obtained from equation (2) by "looking backward" from the current location, X_0 , to the time X_{-1} . In this case Y_1^P is replaced by Y_{-1} which has been evaluated, i.e.,

$$Y_{-1} = Y_0 + A(X_{-1} - X_0) + B(X_{-1} - X_0)^2. \quad (3)$$

Obviously another equation could be written for Y_{-2} but an alternate, more stable and error-free method has been found. Consider the derivative of equation (2) at X_1 , namely,

$$Y_1^{P'} = A + 2B(X_1 - X_0), \quad (4)$$

and the predictor in the form of equation (1),

$$Y_1^{P'} = F_1^P - R_1^P Y_1^P. \quad (5)$$

Substituting the definition of Y_1^P from equation (2) into equation (5) and equating the right hand sides of equation (4) and equation (5) we have

$$A + 2B(X_1 - X_0) = F_1^P - R_1^P [Y_0 + A(X_1 - X_0) + B(X_1 - X_0)^2]. \quad (6)$$

Provided F_1^P and R_1^P are known, equation (6) provides the second equation required to determine A and B .

Let us now consider how F_1^P and R_1^P are determined. Figure 2 shows the current and three previous discrete values of the formation element for a given dependent variable. A parabola of the form

$$F(X) = C_1 + C_2(X - X_0) + C_3(X - X_0)^2, \quad (7)$$

is passed through these four points. The C_j 's ($j=1,2,3$) of equation (7) are found from a least squares fit where the function to be minimized with respect to the C_j is

$$\sum_i W_i [F_i - F(X_i)]^2, \quad -3 \leq i \leq 0. \quad (8)$$

The W_i are weighting functions, chosen so that periodic fluctuations in the F_i values will not propagate into F_1^P . (The relative weights are determined by adding a quantity $\alpha(-1)^i$ to each of the four F_i and requiring that the identical F_1^P be found.) R_1^P is found in a similar fashion. Once F_1^P and R_1^P are determined, A and B can be determined, and consequently, Y_1^P .

The corrector is given by

$$Y_1^C = A_2 Y_{-2} + A_1 Y_{-1} + A_0 Y_0 + (X_1 - X_0)(DY_{-1}' + BY_0' + CY_1^P), \quad (9)$$

where D and B are preselected to minimize both truncation error and noise amplification, and to maximize relative stability. Values for the coefficients in equation (9) can be found in Reference 3.

EXAMPLES. We shall now consider three examples of the types of problems the K-method has been called upon to handle. They are all drawn from the field of aeronomy. Figure 3 shows a linear plot of a piecewise continuous driving function $[Q(t), \text{solid line}]$. The discontinuities are instantaneous since they were formed by reading from a DATA block with (J) and $(J + 1)$ subscripts interchanged. (The reverse of each of the slopes in Figure 3 gives the desired driving function, which is still discontinuous in the first derivative.) The dashed lines are the response of the electron density and a primary positive ion density, here the nitric oxide ion, as a function of time. The curves have been vertically displaced for ease in reading. It is seen that the K-method enables the dependent variables to follow the input discontinuities of the driving term. (Departures from a perfect matching of the input slopes can be explained by chemistry competing with the driving term.) This example demonstrates the effect of careful monitoring of the truncation error.

The second example, Figure 4, shows histograms of the number of species that lie in the decade interval, h/τ_i , where h is the local step size and where τ_i is the instantaneous characteristic time constant of the i th species.¹ (The total area under each histogram corresponds to 64 species.) The numbers to the far right are the decade model times in seconds (i.e., computer execution time runs from bottom to top in this figure). The histograms are divided into a stiff segment to the right of the dashed line, and a non-stiff segment to the left. On the first plot (10^{-5} seconds) only a few species are stiff while at the upper limit of this integration (10^3 seconds) the number has significantly increased, with stiffness factors (h/τ_i) greater than 10^7 .

The last example is shown in Figures 5 and 6. Figure 5 shows the log of the input or driving function, $q(e)$, plotted against the log of time. In the interval 10^{-4} - 10^{-2} seconds those negatively charged species more strongly coupled to the driving function, (e.g. e^- , O_2^- or O_3^-) follow the discontinuities exhibited by the driving term. These details tend to be "washed-out" in the species that are weakly coupled to the driving term (e.g. CO_3^- or CO_4^-). Near 10^2 seconds the strongly coupled species again track the discontinuities in $q(e)$. The dynamic range in the dependent variables is about six orders of magnitude. Figure (6) shows the broad range response of the neutral species. This graph shows those species that follow the driving term [e.g. $O(^1D)$, $N(^2D)$], those that are independent of the driving term [e.g. N_2O , CO] and those that tend to be chemistry dominated [e.g. H_2 , HNO_2 , N_2O_5].

In summary, we have empirically derived a third order, variable step size method for efficiently handling stiff ODEs that appears to work for discontinuous driving functions. We anticipate with some further work that this method will be put on a firmer mathematical foundation.

REFERENCES

1. Numerical Methods, by G. Dahlquist and A. Björck, Trans. by N. Anderson, 1974, Prentice-Hall, Inc., Englewood Cliffs, NJ, p. 22.
2. See for example Numerical Methods for Scientists and Engineers, by R. W. Hamming, 2nd Edition, 1973, McGraw-Hill, Inc. p. 5.
3. An earlier version has been reported. See "Description and Comparison of the K-Method for Performing Numerical Integration of Stiff Ordinary Differential Equations," by M. D. Kregel and E. L. Lortie, BRL Report No. 1733, July 1974, AD #A003855.

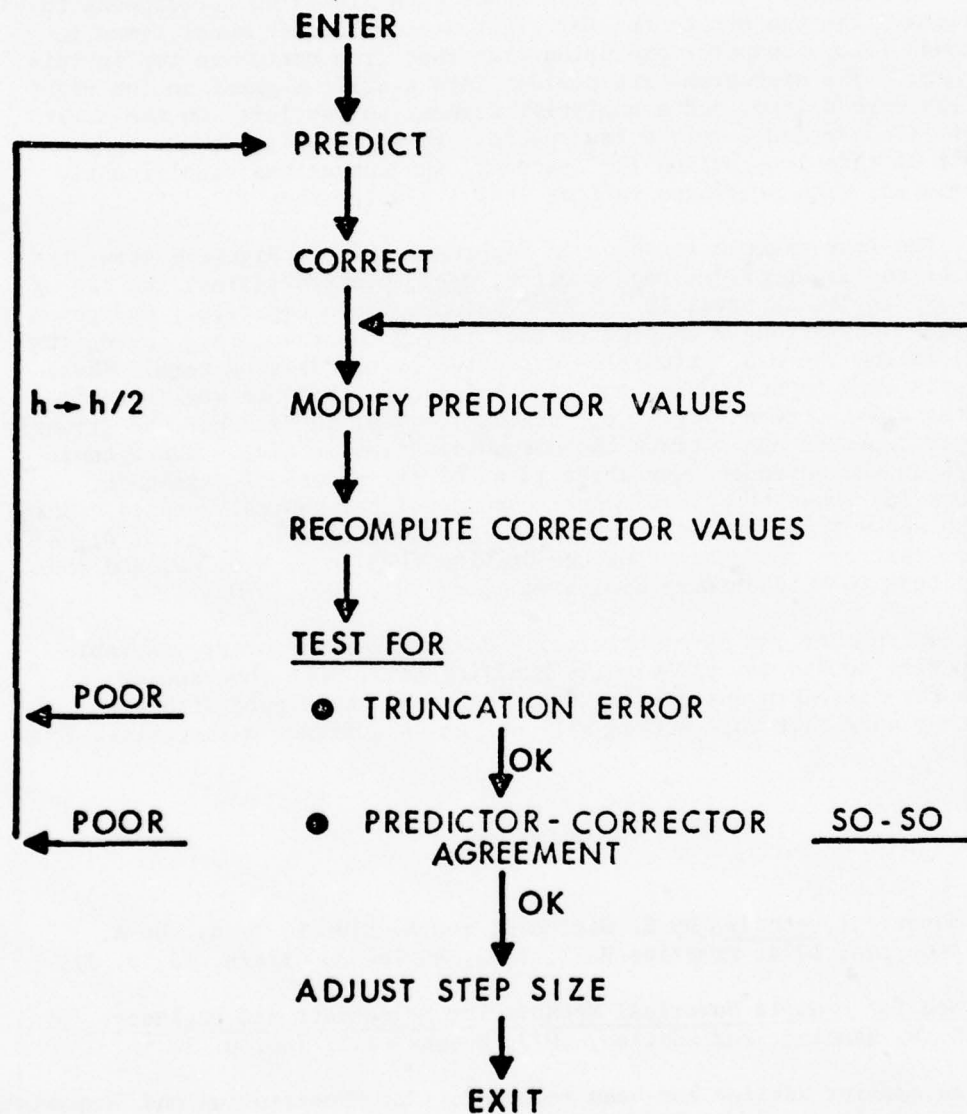


Figure 1. Schematic of K-method of integration; h is the step size.

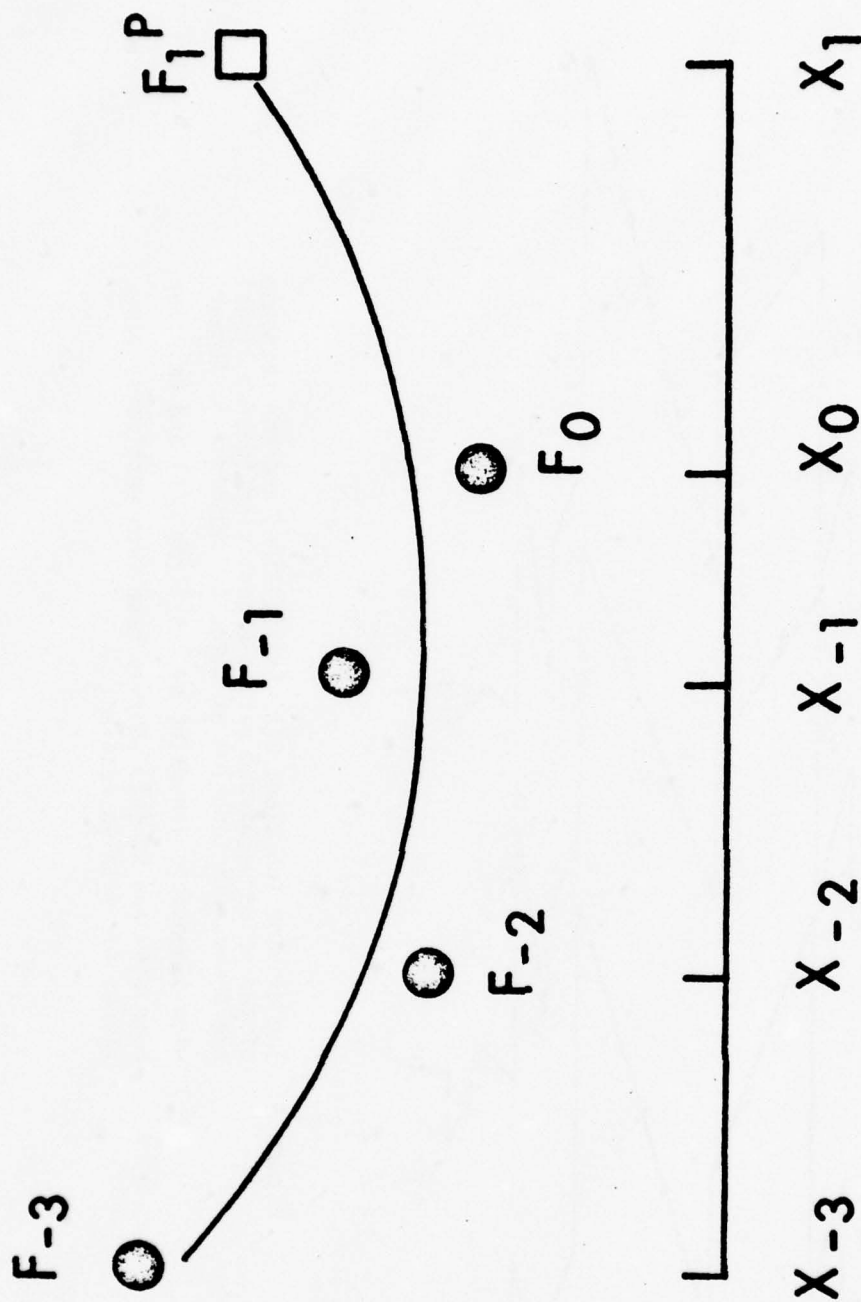


Figure 2. Schematic of a parabolic fit to the values of F at the current and three previous times for one of the dependent variables. F_1^P is found by continuing this parabola to the time X_1 .

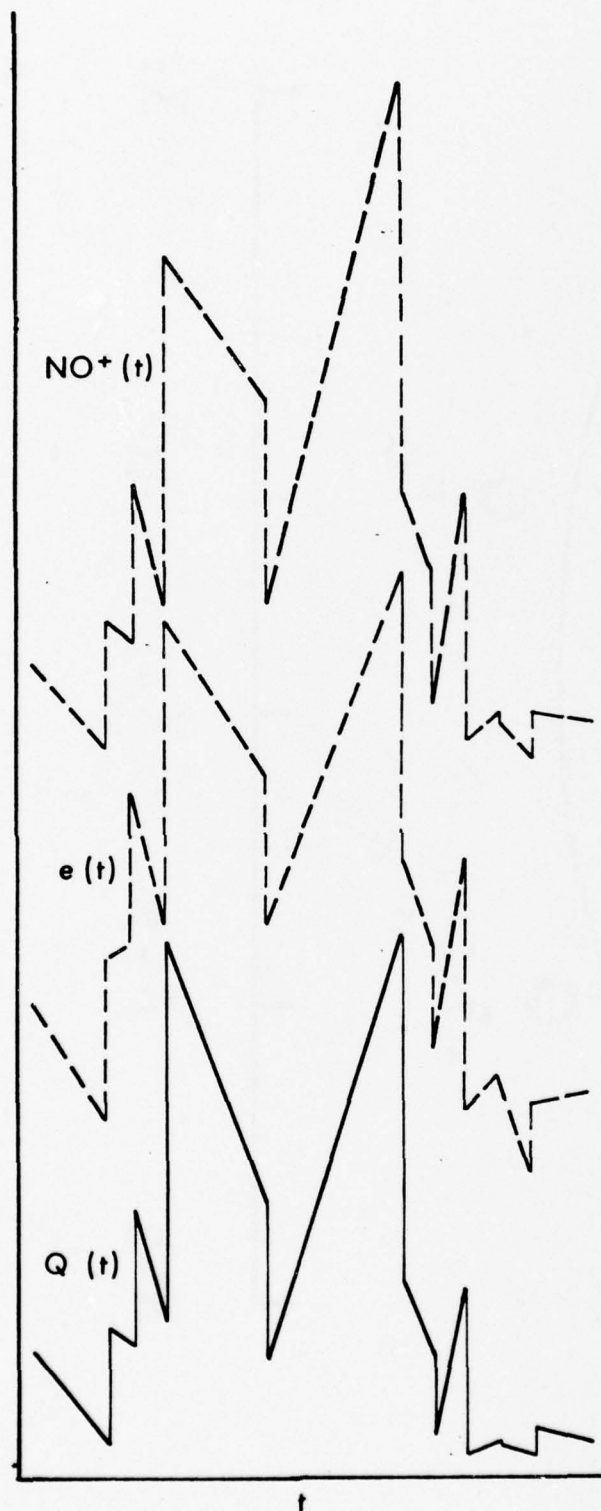


Figure 3. The driving function, $Q(t)$ (solid line) and the response of two primary charged particle densities (dashed lines) are shown as a function of time, t . Abscissa is linear with dimensions ion-pairs $\text{cm}^{-3} \text{s}^{-1}$ for $Q(t)$ and cm^{-3} for both $e(t)$ and $\text{NO}^+(t)$. Curves have been vertically displaced for ease in reading.

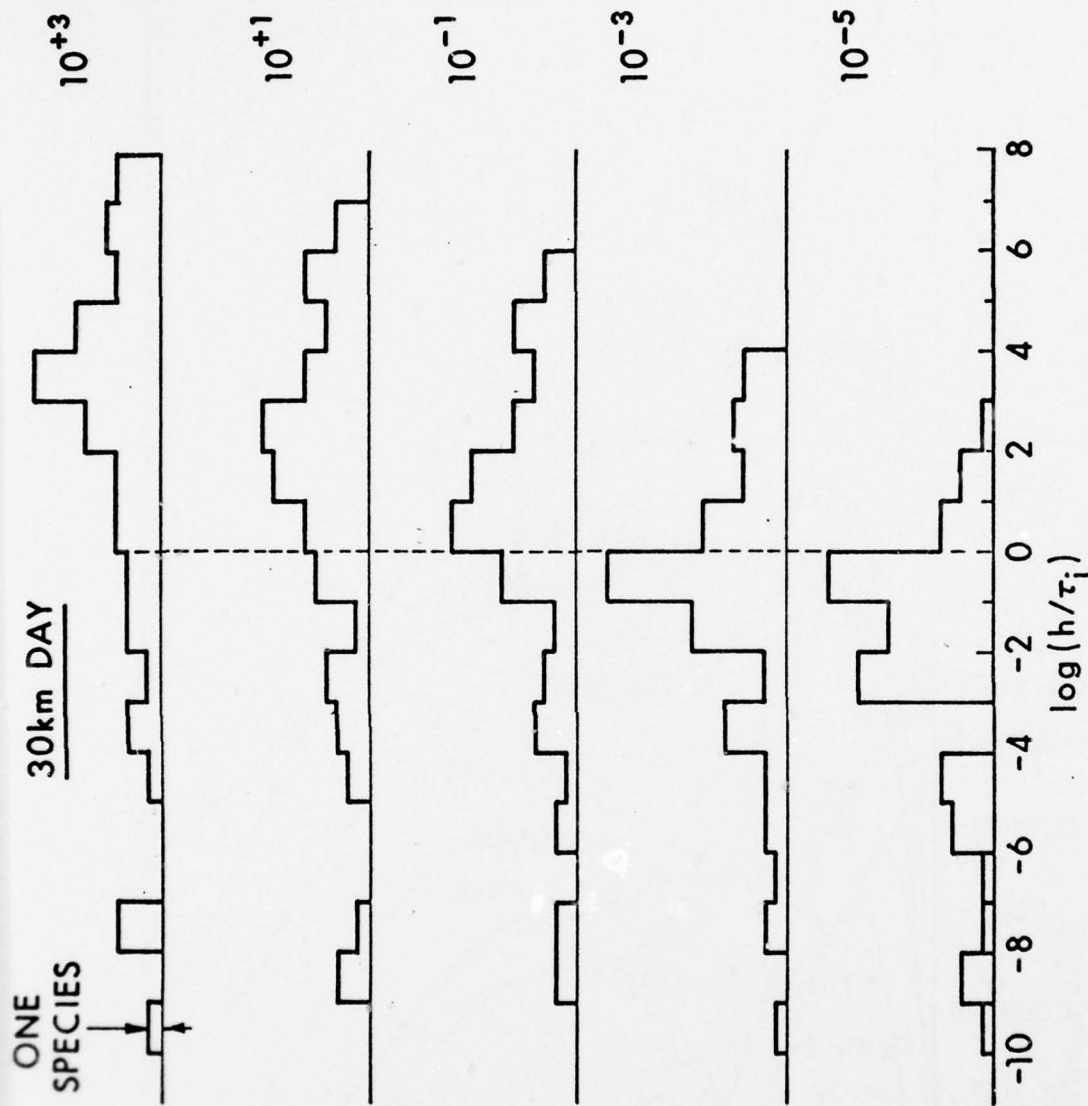


Figure 4. Histograms that characterize the stiffness of chemical species found in a daytime deionization model of the atmosphere at an altitude of 30 km. (See text for details).

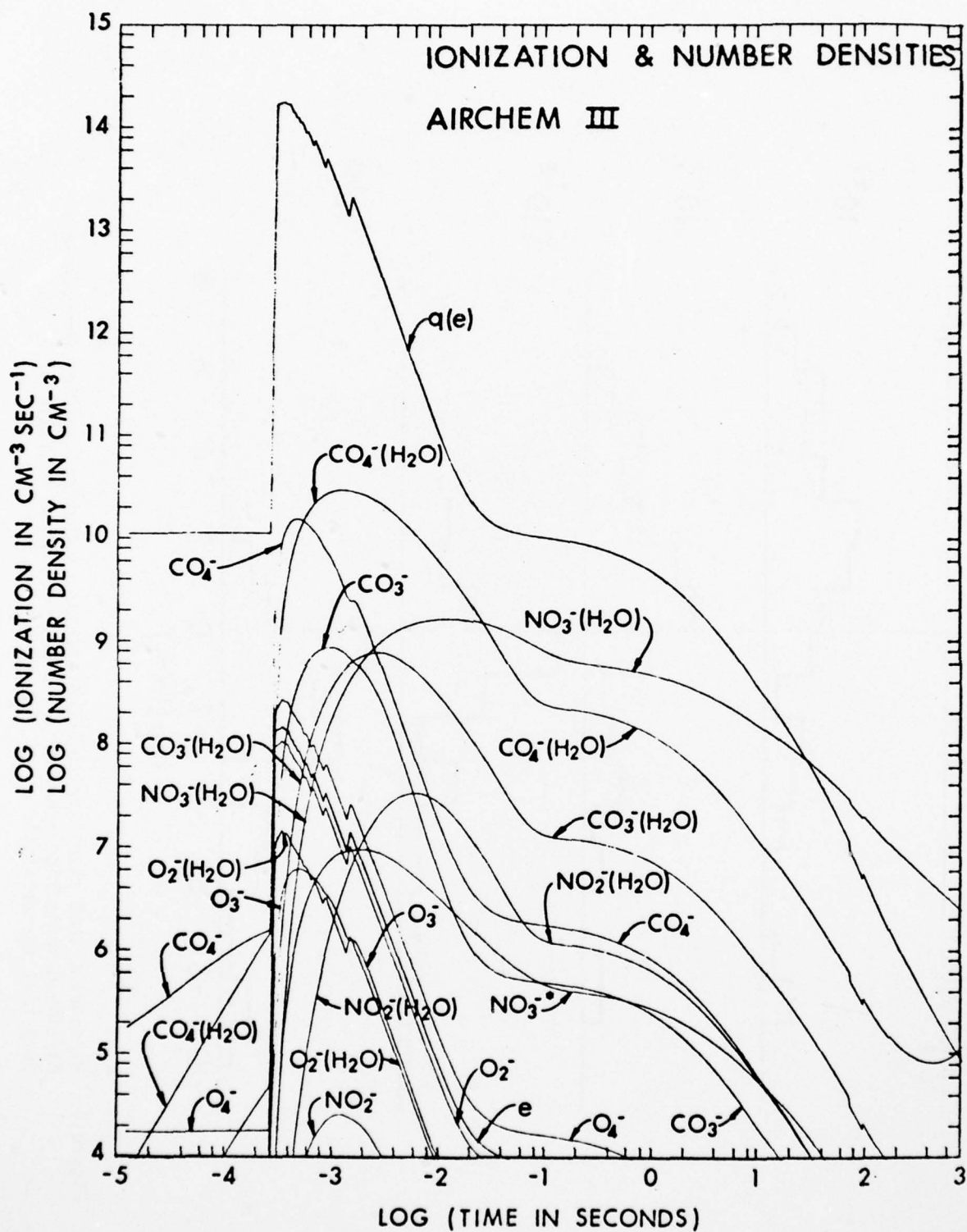


Figure 5. Modeled response of the negative ion densities to the driving function $q(e)$.

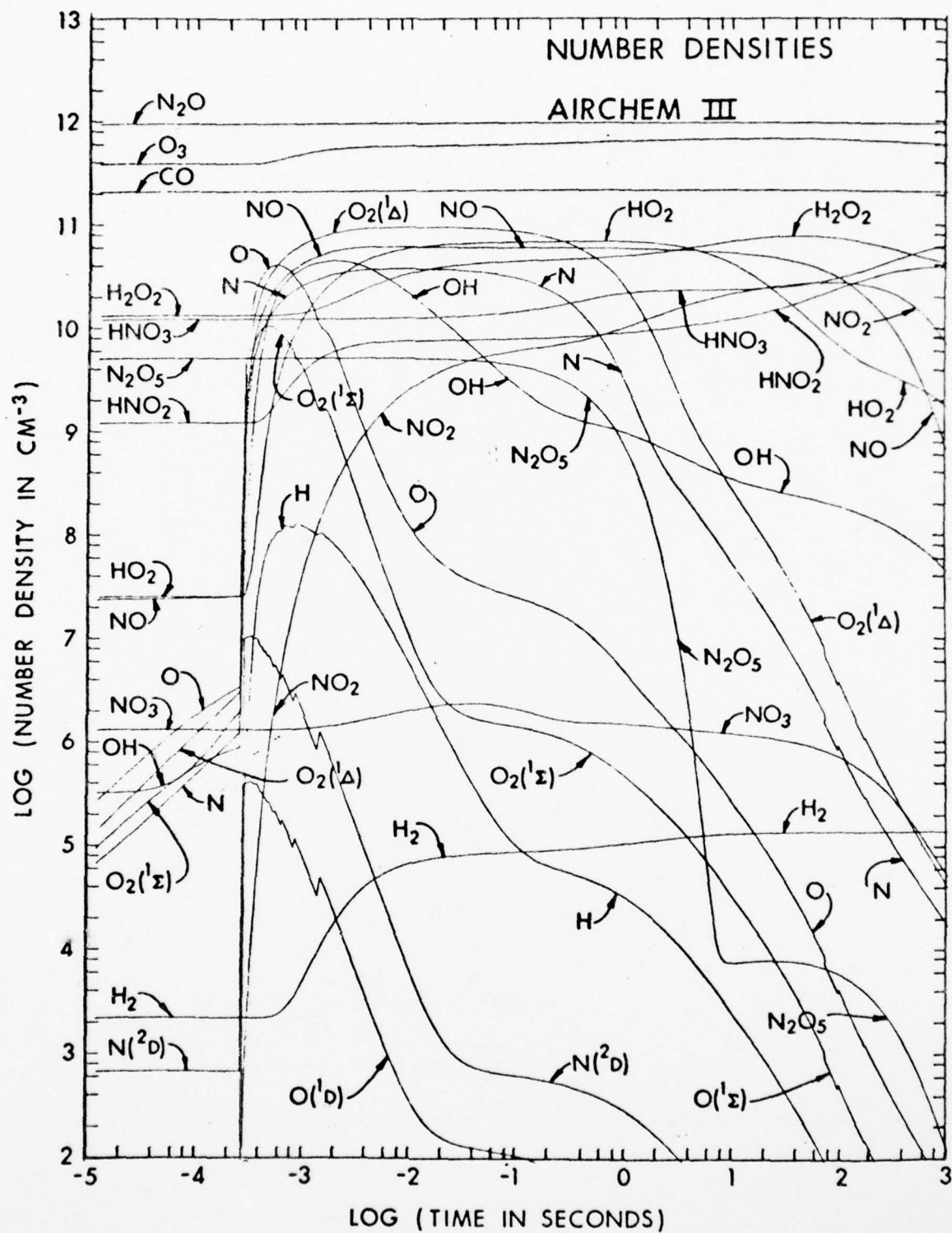


Figure 6. Modeled response of the neutral densities to the same driving function of Fig. 5.

A_0 -STABILITY OF BROWN'S MULTISTEP MULTIDERIVATIVE METHODS *

Rolf Jeltsch
Institute of Mathematics
Ruhr-University Bochum
D- 4630 Bochum, Germany

ABSTRACT. Brown [1] introduced k -step methods using ℓ derivatives. Necessary and sufficient conditions for A_0 -stability and stiff stability of these methods are given. These conditions are used to investigate for which k and ℓ the methods are A_0 -stable. It is seen that for all k and ℓ with $k \leq 1.5(\ell + 1)$ the methods are A_0 -stable and stiffly stable. This result is conservative and can be improved for ℓ sufficiently large. For small k and ℓ A_0 -stability has been determined numerically by implementing the necessary and sufficient condition.

1. INTRODUCTION. To solve numerically the initial value problem

$$(1) \quad y'(x) = f(x, y(x)) \quad , \quad y(a) = \eta$$

we shall need the higher derivatives $f^{(j)}(x, y)$, which are defined by $f^{(0)}(x, y) = f(x, y)$ and

$$f^{(j)}(x, y) = \frac{\partial}{\partial x} f^{(j-1)}(x, y) + \frac{\partial}{\partial y} f^{(j-1)}(x, y) f(x, y) \quad , \quad j = 1, 2, \dots$$

Let $h > 0$ be the stepsize, $x_m = a + mh$, and y_m be approximations to the exact solution $y(x_m)$ of (1). Moreover let $f_m^{(j)} = f^{(j)}(x_m, y_m)$. Brown [1] has introduced methods of the form

$$(2) \quad \sum_{i=0}^k \alpha_i y_{n+i} = \sum_{j=1}^{\ell} h^j \beta_j f_{n+k}^{(j-1)} \quad \text{with} \quad \sum_{i=0}^k |\alpha_i| > 0 \quad , \quad \sum_{j=1}^{\ell} |\beta_j| > 0 \quad ,$$

$n = 0, 1, 2, \dots$. Here α_i and β_j are constants such that the method has highest possible error order. In Jeltsch, Kratz [13] it was shown that

$$(3a) \quad \alpha_i = (-1)^{k-i} \binom{k}{i} (k-i)^{-\ell} \quad , \quad i = 0, 1, \dots, k-1$$

$$(3b) \quad \beta_j = (-1)^j / j! \sum_{i=0}^{k-1} (-1)^{k-i} \binom{k}{i} (k-i)^{j-\ell} \quad , \quad j = 0, 1, \dots, \ell$$

Here we have used the natural convention that $\alpha_k = -\beta_0$. A method of form (2) is said to have error order p if

$$\sum_{i=0}^k \alpha_i y(x+ih) - \sum_{j=1}^{\ell} h^j \beta_j y^{(j)}(x+kh) = C_{p+1} h^{p+1} y^{(p+1)}(x) + O(h^{p+2}) \quad , \quad C_{p+1} \neq 0 \quad ,$$

for all sufficiently often differentiable functions $y(x)$. In [13] it was shown that the methods given by (2) and (3) have the error order $p = k + \ell - 1$ and that there is no method of form (2) with a higher error order. We shall call the formula (2) with (3) Brown's (k, ℓ) -method. It was discussed in [13] for which k and ℓ Brown's (k, ℓ) -method is stable and for which it is not. Since the method belongs to the class of k -step methods with ℓ derivatives and $p \geq 1$ it converges if and only if it is stable (see e.g. Griepentrog [8], Spijker [15]). Convergence is also shown for strongly stable methods in Brown [1], [2]. Assuming strong stability an estimate for the global discretization error as well as the first term in the asymptotic error expansion are given in [1], [2]. In the present article we give necessary and sufficient conditions for the methods to be A_0 -stable. These conditions are then used to investigate for which k and ℓ Brown's (k, ℓ) -methods are A_0 -stable, see Fig. 1. For k and ℓ small this has been done by implementing the criterias on the computer. It is proved that Brown's

* This article has been supported by the European Research Office.

(k, ℓ) -methods are A_0 -stable and stiffly stable if $k \leq 1.5(\ell + 1)$. For any $\alpha < \alpha_1 \sim 3.59112148$ it is A_0 -stable and stiffly stable for $k \leq \alpha(\ell + 1)$ and ℓ sufficiently large. Computer results suggest that $\ell \geq 2$ is already large enough. Finally we would like to mention that Brown's (k, ℓ) -methods are the well known Backward Differentiation Methods BDM. In Liniger [14] it has been shown that the BDM are A_0 -stable and strongly stable for $k \leq 5$. Gear [7] showed by plotting the boundary of the region of absolute stability that the BDM are stiffly stable for $k \leq 6$. It is well known, see Cryer [4], Creedon, Miller [3] that the BDM are unstable for $k > 6$.

In the next section we derive necessary and sufficient conditions for A_0 -stability of methods of form (2). In Section (3) these conditions are used to derive our main results on Brown's (k, ℓ) -methods while the more technical proofs are postponed to the last section.

2. Necessary and sufficient conditions for A_0 -stability. Applying the method (2) to $y' = \lambda y$ leads to the recurrence relation

$$(3) \quad \sum_{i=0}^k \alpha_i y_{n+i} - \sum_{j=1}^{\ell} \beta_j (h\lambda)^j y_{n+k} = 0, \quad n = 0, 1, \dots$$

This is a linear homogeneous recursion relation with constant coefficients and thus its characteristic equation is

$$(4) \quad \Phi(z, u) = \rho(z) - u z^k n(u) = 0.$$

Here we have used the abbreviation $u = h\lambda$ and the polynomials

$$(5) \quad \rho(z) = \sum_{i=0}^k \alpha_i z^i, \quad n(u) = \sum_{j=1}^{\ell} \beta_j u^j.$$

Let $z_i(u)$, $i = 1, 2, \dots, k$ be the k branches of the algebraic function $z(u)$ defined by

$$(6) \quad \Phi(z(u), u) = 0.$$

Then every solution of (3) can be written in the form

$$(7) \quad y_n = \sum_{j=1}^n \pi_j(n) z_j^n(u),$$

where $\pi_j(n)$ are polynomials in n with constant coefficients whose degree is less than the multiplicity of $z_j(u)$ as a root of $\Phi(z, u)$. * in (7) indicates that the sum is only taken over those j for which the branch $z_j(u)$ is finite. Clearly $z_i(0)$ are the roots of $\rho(z)$. A method is called stable or more precisely Dahlquist stable if $|z_i(0)| \leq 1$ for $i = 1, 2, \dots, k$ and if $|z_i(0)| = 1$ then $z_i(0)$ is a simple root of $\rho(z)$, see e.g. Henrici [9], Stetter [16, p. 206]. A method is convergent if it is stable and the error order p exceeds 0, Griepentrog [8], Spijker [15]. It is well known that for a convergent method there is exactly one branch of $z(u)$ which becomes 1 at $u = 0$. We call this branch the principal branch and denote it by $z_1(u)$. Hence $z_1(0) = 1$ and $z_j(0) \neq 1$ for $j = 2, 3, \dots, k$. A method is called strongly stable if

$$(8) \quad |z_i(0)| < 1, \quad i = 2, 3, \dots, k.$$

In the weakstability analysis of methods one is interested in the asymptotic behaviour of y_n given by (7) as n tends to infinity and h is kept fix. Hence one is interested in the region of absolute stability

$$(9) \quad A = \{u \in \mathbb{C} \mid |z_i(u)| < 1, \quad i = 1, 2, \dots, k\}.$$

According to Cryer [5] a method is called A_0 -stable if A contains the negative real axis, i.e. $(-\infty, 0) \subset A$. Gear [7] introduced the concept of stiff stability which we shall use in the rigorous form given in Jeltsch [11], [12]. Since $\zeta_1(0)$ is a simple root of $\rho(\zeta)$ the principal branch $\zeta_1(u)$ of the algebraic function $\zeta(u)$ is analytic in a neighborhood of $u = 0$. Let Ω be the largest star region into which the principal branch has an analytic continuation. The set

$$(10) \quad R = \{u \in \Omega \mid |\zeta_i(u)| < |\zeta_1(u)|, \quad i = 2, 3, \dots, k\}$$

is called the region of relative stability. Let D , θ and a be positive constants and define the sets $R_1 = \{u \in \mathbb{C} \mid \operatorname{Re} u < -\theta\}$, $R_2 = \{u \in \mathbb{C} \mid \operatorname{Re} u \leq -a, \operatorname{Im} u < \theta\}$ and $R_3 = \{u \in \mathbb{C} \mid |\operatorname{Re} u| < a, \operatorname{Im} u < \theta\}$. A method is called stiffly stable if it is convergent A_0 -stable and if for some positive D , θ and a one has $R_1 \cup R_2 \subset A$ and $R_3 \subset R$.

THEOREM 1. A method of form (2) with an error order $p \geq 1$ is stiffly stable if and only if it is A_0 -stable and strongly stable.

This theorem follows immediately from Theorem 3 in Jeltsch [12]. The above theorem is one important reason why we are interested in finding necessary and sufficient conditions for A_0 -stability. In the following we shall derive such conditions for methods of form (2) which satisfy

$$(11a) \quad (-1)^{j+1} \beta_j \geq 0, \quad j = 0, 1, \dots, p$$

$$(11b) \quad \alpha_k = -\beta_0 + \theta, \quad \beta_p \neq 0$$

$$(11c) \quad \rho(1) = 0.$$

However these methods need not to be convergent. We introduce the standard transformation

$$(12) \quad z = \frac{\zeta+1}{\zeta-1}, \quad \zeta = \frac{z+1}{z-1},$$

which maps the unit disk of the ζ -plane onto the left half plane $H^- = \{z \in \mathbb{C} \mid \operatorname{Re} z < 0\}$ of the z -plane. Accordingly we introduce

$$(13) \quad \begin{aligned} \Psi(z, u) &= (z-1)^k \psi\left(\frac{z+1}{z-1}, u\right) \\ &= r(z) - \rho(u) (z+1)^k = 0, \end{aligned}$$

where

$$(14) \quad r(z) = (z-1)^k \rho\left(\frac{z+1}{z-1}\right) = \sum_{i=0}^k a_i z^i.$$

Assume that $\rho(\zeta)$ has a root $\zeta = 1$ of exact multiplicity m . Because of (11c) one has $m \geq 1$. Clearly in most applications one will have $m = 1$. Since the transformation (12) maps $\zeta = 1$ into $z = \infty$ the polynomial $r(z)$ is of exact degree $k - m$. Let $z(u)$ be the algebraic function defined by $\Psi(z(u), u) = 0$ and $z_i(u)$, $i = 1, 2, \dots, k$, its branches. Clearly a method is A_0 -stable if $\operatorname{Re} z_i(u) < 0$ for all $u \in (-\infty, 0)$ and all $i = 1, 2, \dots, k$. Dividing (13) by $(z+1)^k$ one finds immediately that $z_i(\infty) = -1$ for $i = 1, 2, \dots, k$. From (11) follows that $\operatorname{Im} z_i(u) < 0$ for all $u \in (-\infty, 0)$ and therefore $\Psi(z, u)$ is a polynomial in z of exact degree k for any $u \in (-\infty, 0)$. Hence $z_i(u)$ is always finite for all $u \in (-\infty, 0)$ and all $i = 1, 2, \dots, k$. The branches $z_i(u)$ can be defined such that these are functions of $u \in (-\infty, 0)$. Hence the method is A_0 -stable if and only if none of the graphs $z_i(u)$ intersects the imaginary axis for $u \in (-\infty, 0)$. Hence we have shown the following

BEST AVAILABLE COPY

LEMMA 1. Assume a method of form (2) satisfies (11). Then the method is A_0 -stable if and only if

$$(15) \quad r(iy) - u_n(u) (iy+1)^k \neq 0$$

for all $u \in (-\infty, 0)$ and all $y \in \mathbb{R}$.

Clearly Lemma 1 is not a constructive criteria for A_0 -stability since (15) has to be tested for infinitely many pairs u and y . In order to formulate a constructive criteria for A_0 -stability we introduce the polynomials

$$(16a) \quad P(\tau) := -v^{-1} \operatorname{Im} \{r(iv)(-iy+1)^k\}$$

and

$$(16b) \quad Q(\tau) := \operatorname{Re} \{r(iv)(-iy+1)^k\},$$

where $\tau := y^2$. $P(\tau)$ and $Q(\tau)$ are real polynomials of degree $k-1$. Let $\tau_1, \tau_2, \dots, \tau_s$ be all non negative roots of $P(\tau)$.

THEOREM 2. Assume a method of form (2) satisfies (11). Then the method is A_0 -stable if and only if the following two conditions hold.

$$A_1: \quad Q(0) = r(0) = (-1)^k \rho(-1) \geq 0$$

$$A_2: \quad Q(\tau_j) \geq 0 \quad \text{for } j = 1, 2, \dots, s.$$

Note that a convergent method always satisfies A_1 . Clearly condition A_2 is satisfied if $P(\tau)$ has no non negative roots at all. This gives the following

COROLLARY 1. Assume a method of form (2) satisfies (11). Then the conditions B_1, B_2 are sufficient for A_0 -stability.

$$B_1: \quad r(0) = (-1)^k \rho(-1) \geq 0$$

$$B_2: \quad P(\tau) \neq 0 \quad \text{for all } \tau \in [0, \infty).$$

REMARK. Condition B_1 is easily checked and condition B_2 can be checked in finitely many additions, subtractions, multiplications and divisions using the Sturm algorithm, see Werner [17, p. 132], Dickson [6, p. 83]. B_2 is trivially satisfied if all coefficients of $P(\tau)$ have the same sign.

PROOF OF THEOREM 2. (15) is equivalent to

$$(17) \quad r(iy)(iy+1)^{-k} \neq u_n(u), \quad u \in (-\infty, 0), \quad y \in \mathbb{R}.$$

From (11) follows that $u_n(u)$ is a polynomial which maps $(-\infty, 0)$ onto $(-\infty, 0)$. Hence by Lemma 1 and (17) a method is A_0 -stable if and only if

$$(18) \quad r(iy)(iy+1)^{-k} \notin (-\infty, 0) \quad \text{for all } y \in \mathbb{R}.$$

But by (16) one has

$$(19) \quad \begin{aligned} Q(y^2) - iv P(y^2) &= r(iv)(-iy+1)^k \\ &= r(iv)(iy+1)^{-k} (1+v^2)^k. \end{aligned}$$

Hence one has A_0 -stability if and only if

$$(20) \quad Q(y^2) - iy P(y^2) \neq (-\infty, 0) \text{ for all } y \in \mathbb{R}.$$

(20) is now obviously equivalent to the conditions A_1 and A_2 . \square

We would like to recall that in Theorem 2 we did not assume that the method is stable or has an error order $p \geq 1$. The reason is that through slight modifications of the conditions A_1, A_2 one obtains a criteria which gives A_0 -stability and strong stability.

THEOREM 3. Assume a method of form (2) satisfies (11) and has an error order $p \geq 1$. Then the method is A_0 -stable and strongly stable if and only if the following three conditions hold.

$$\begin{aligned} C_1: & \quad a_0 = r(0) = (-1)^k \rho(-1) > 0 \\ C_2: & \quad 0(\tau_j) > 0 \text{ for } j = 1, 2, \dots, s \\ C_3: & \quad a_{k-1} > 0. \end{aligned}$$

PROOF. (I) Necessity. Assume that the method is A_0 -stable and strongly stable. Hence A_1 and A_2 hold. Since $z = -1$ is not a root of $\rho(z)$ one has C_1 . Strong stability together with $p \geq 1$ implies that $z = 1$ is a simple root of $\rho(z)$, that is $m = 1$. As we have seen earlier this implies $a_{k-1} \neq 0$. However since $r(z)$ is a polynomial with all roots in the left hand plane H^- we have that a_0 and a_{k-1} have the same sign. This implies C_3 . We have already seen that $z_i(u)$ is continuous for $u \in [-\infty, 0)$ for $i = 1, 2, \dots, k$. Since $a_k = 0$ and $a_{k-1} \neq 0$ one of these roots $z_i(u)$, let us denote it by $z_1(u)$, will tend to infinity as u tends to 0^- while the other ones remain finite. Hence

$$(21) \quad \lim_{u \rightarrow 0^-} z_i(u) = z_i(0) \quad i = 2, 3, \dots, k$$

are the roots of $r(z)$. Strong stability implies that $\operatorname{Re} z_i(0) < 0$. Hence (15) has to hold for all $u \in (-\infty, 0]$ and all $y \in \mathbb{R}$. As in the proof of Theorem 2 this implies that

$$(22) \quad 0(y^2) - iy P(y^2) \neq (-\infty, 0] \text{ for all } y \in \mathbb{R}.$$

From (22) follows immediately C_2 .

(II) Sufficiency. From C_1 and C_2 follows by Theorem 1 that the method is A_0 -stable. C_3 ensures that $\rho(z)$ has a simple root at $z = 1$. Hence we find as in (I) that (21) holds. From C_1 and C_2 follows that (22) holds and hence $\operatorname{Re} z_i(u) < 0$ for all $u \in (-\infty, 0]$ and $i = 2, 3, \dots, k$. In particular $\operatorname{Re} z_i(0) < 0$ for $i = 2, 3, \dots, k$. Hence the method is strongly stable. \square

REMARK 1. Using Theorem 1 we see that conditions C_1 - C_3 are necessary and sufficient for a method of form (2) with (11) and $p \geq 1$ to be stiffly stable.

2. Note that a convergent method always satisfies C_1 and C_3 .

Condition C_2 is satisfied if $P(\tau)$ has no non negative root. This gives the

COROLLARY 2. Assume a method of form (2) satisfies (11) and has $p \geq 1$. Then D_1 - D_3 are sufficient for A_0 -stability, strong stability and stiff stability.

$$\begin{aligned} D_1: & \quad a_0 = r(0) > 0 \\ D_2: & \quad P(\tau) \neq 0 \text{ for all } \tau \in [0, \infty) \end{aligned}$$

$$D_3: \quad a_{k-1} > 0.$$

As we shall see in the next section Corollary 2 will be very powerful for the investigation of Brown's (k, ℓ) -methods. For completeness we give the following result.

THEOREM 4. Assume a method of form (2) satisfies (11) and has an error order $p \geq 1$. Then the method is A_0 -stable and Dahlquist stable if and only if the conditions E_1 - E_4 hold.

$$E_1: \quad a_0 = r(0) \geq 0,$$

$$E_2: \quad Q(\tau_j) \geq 0 \text{ for } j = 1, 2, \dots, s,$$

$$E_3: \quad a_{k-1} > 0,$$

$$E_4: \quad \text{If } 0(\tau_j) = 0 \text{ then } z = \pm i \tau_j^{1/2} \text{ is a simple root of } r(z).$$

The proof goes along the lines of the proofs of Theorem 2 and 3 and is therefore left to the reader.

3. A_0 -stability of Brown's (k, ℓ) -method. In this section we shall investigate for which k and ℓ Brown's (k, ℓ) -method is A_0 -stable. It is clear that if a method is unstable due to a root of $\rho(z)$ which lies outside the unit circle then the method is also not A_0 -stable. Hence from Jeltsch, Kratz [13] follows that to any fix ℓ Brown's (k, ℓ) -method is not A_0 -stable for all k large enough. However if we fix k then Brown's (k, ℓ) -method becomes A_0 -stable, strongly stable and stiffly stable for all ℓ large enough. To show this let us first show that Brown's (k, ℓ) -method satisfies (11). In [13] it was shown that the following relation holds

$$(23a) \quad \sum_{i=0}^{k-1} (-1)^{k-1-i} \binom{k}{i} (k-i)^{j-\ell} = \sum_{t_1=1}^k \frac{1}{t_1} \sum_{t_2=1}^{t_1-1} \frac{1}{t_2} \dots \sum_{t_{\ell-j-1}=1}^{t_{\ell-j-2}-1} \frac{1}{t_{\ell-j-1}} \sum_{t_{\ell-j}=1}^{t_{\ell-j-1}-1} \frac{1}{t_{\ell-j}} > 0, \quad j=0, 1, \dots, \ell-1.$$

For $j = \ell$ one finds through direct calculation

$$(23b) \quad \sum_{i=0}^{k-1} (-1)^{k-1-i} \binom{k}{i} = 1 > 0.$$

Using (23) in (3b) one finds that (11a) and (11b) hold. Note that we could have established (11a) and (11b) also from the fact that Brown's methods are Hermite interpolatory, see Theorem 9 in [10]. The assumption (11c) is satisfied since Brown's methods have the error order $p = k + \ell - 1 \geq 1$.

THEOREM 5. Brown's (k, ℓ) -method is A_0 -stable, and strongly stable whenever

$$(24) \quad k \leq \frac{3}{2} (\ell + 1).$$

PROOF. We show this using Corollary 2. From (14) and (3) one obtains

$$(25) \quad r(z) = \sum_{i=0}^{k-1} (-1)^{k-i} \binom{k}{i} (k-i)^{-\ell} ((z+1)^i (z-1)^{k-i} - (z+1)^k).$$

Hence

$$(26) \quad a_0 = \sum_{i=0}^{k-1} \binom{k}{i} (k-i)^{-\ell} (1 - (-1)^{k-i}) > 0$$

and

$$(27) \quad a_{k-1} = 2 \sum_{i=0}^{k-1} (-1)^{k-1-i} \binom{k}{i} (k-i)^{1-\ell}.$$

(26) implies that D_1 holds and (27) together with (23) show that D_3 holds. In the remaining part of the proof we shall show that D_2 holds whenever (24) is satisfied. From (25) and (16a) we find through an easy but tedious computation that

$$(28) \quad P(\tau) = 2 \sum_{j=0}^{k-1} (-1)^{k-1-j} \binom{k}{j} (k-j)^{-\ell} (1+\tau)^j \sum_{t=0}^{\ell} \binom{k-j}{2t+1} (-1)^{k-1-j-2t} 4^t (-\tau)^t,$$

with

$$(29) \quad \ell := \lfloor (k-1-j)/2 \rfloor.$$

Here $\lfloor a \rfloor$ denotes the largest integer not exceeding a . In order to show that $P(\tau)$ has no non negative roots we have to distinguish the following cases

I. $\ell \leq 7$. Direct calculation on a TR 440 in double precision (i.e. 81-84 bit-mantissa) showed that all coefficients of $P(\tau)$ are positive whenever k satisfies (24). Hence $P(\tau) > 0$ for all $\tau \in [0, \infty)$ and hence D_2 is satisfied.

II. $\ell > 7$. We split $P(\tau)$ as follows

$$(30) \quad \frac{1}{2} P(\tau) = S(\tau) + T(\tau),$$

where

$$(31) \quad S(\tau) = \sum_{j=0}^{k-1} (-1)^{k-1-j} \binom{k}{j} (k-j)^{1-\ell} (1+\tau)^j (1-\tau)^{k-1-j}$$

and

$$(32) \quad T(\tau) = \sum_{j=0}^{k-3} (-1)^{k-1-j} \binom{k}{j} (k-j)^{-\ell} (1+\tau)^j \sum_{t=1}^{\ell} \binom{k-j}{2t+1} (-1)^{k-1-j-2t} 4^t (-\tau)^t,$$

where ℓ is given by (29). We shall need the following two lemmata.

LEMMA 2. Let $1 \leq k \leq 3.3(\ell+1)$ and $\ell \geq 8$. Then one has

$$(33) \quad S(\tau) > k(1+\tau)^{k-2} \quad \text{for all } \tau \in [0, \infty).$$

Moreover let $\alpha < \alpha_1$, where α_1 is the unique positive root of

$$(34) \quad q(\alpha) = 1$$

with

$$(35) \quad q(\alpha) := \alpha e^{-1 - \frac{1}{\alpha}}.$$

Then (33) holds for ℓ sufficiently large. α_1 is approximately 3.59112148.

LEMMA 3. Let $3 \leq k \leq 1.5(\ell+1)$ and $\ell \geq 8$. Then one has

$$(36) \quad |T(\tau)| \leq \frac{1}{4} k(k-1)(k-2)^2 3^{2-\ell} (1+\tau)^{k-2} \quad \text{for all } \tau \in [0, \infty).$$

In order not to interrupt the idea of the proof we postpone the proof to Section 4. For $k = 1, 2$ one has from (32) that $T(\tau) = 0$. Hence by Lemma 2 and (30) one has $P(\tau) > 0$ for all $\tau \in [0, \infty)$. Thus condition D_2 holds. In the following let $k \geq 3$. From (30), (33) and (36) follows

$$\frac{1}{2} P(\tau) > S(\tau) - |T(\tau)| > (\tau+1)^{k-2} \left(k - \frac{1}{4} k(k-1)(k-2)^2 3^{2-\ell} \right)$$

for all $\tau \in [0, \infty)$, $3 \leq k \leq 1.5(\ell+1)$, $\ell \geq 8$. Hence $P(\tau)$ has no non negative root if

$$(37) \quad 3^\ell > \frac{9}{4} (k-1)(k-2)^2.$$

Since the right hand side of (37) is monotonically increasing in k it is enough to show (37) for $k = 1.5 (\ell + 1)$. Hence it remains to show that

$$(38) \quad 3^\ell > \frac{9}{32} (3\ell + 1)(3\ell - 1)^2.$$

However it is easy to see that (38) holds for $\ell \geq 8$.

Collecting the results from I and II we find that $P(\tau) > 0$ for all $\tau \in [0, \infty)$ for $1 \leq k \leq 1.5 (\ell + 1)$ and $\ell = 1, 2, \dots$. Hence D_2 is satisfied. \square

Clearly in the proof of (24) we have made some conservative estimates. We can improve upon (24) if we consider the methods only for ℓ large enough.

THEOREM 6. Let $\alpha < \alpha_1$ where α_1 is the unique positive root given by (34) and (35), $\alpha_1 \sim 3.59112148$. Then Brown's (k, ℓ) -method is A_0 -stable and strongly stable for all k with

$$(39) \quad k \leq \alpha(\ell + 1)$$

and ℓ sufficiently large.

As we shall see from Table 1 computer results suggest that for all $\alpha \leq \alpha_1$ it is in fact in Theorem 6 enough to take $\ell \geq 2$.

PROOF OF THEOREM 6. As in the proof of Theorem 5 we use Corollary 2. We have already seen that D_1 and D_3 are always satisfied. In order to show D_2 we split $P(\tau)$ as in (30) and need estimates for $S(\tau)$ and $T(\tau)$. For $\alpha < \alpha_1$ Lemma 2 gives the lower bound (33) for ℓ sufficiently large. The following lemma gives an upper bound for $T(\tau)$.

LEMMA 4. Let $\alpha \in (1, \infty)$. Then there exists $L_\alpha > 0$ such that

$$(40) \quad |T(\tau)| \leq \frac{9}{4} \alpha^4 \frac{4}{\alpha} 3^{-\ell} (1+\tau)^{k-2} \quad \text{for } \tau \in [0, \infty)$$

$k \leq \alpha\ell$ and $\ell \geq L_\alpha$.

We postpone the proof to section 4. Let $\alpha < \alpha_1$ then using (30), (33) and (40) one has

$$\frac{1}{2} P(\tau) > S(\tau) - |T(\tau)| > (1+\tau)^{k-2} \left\{ k - \frac{9}{4} \alpha^4 \frac{4}{\alpha} 3^{-\ell} \right\}$$

for $\tau \in [0, \infty)$, $1 \leq k \leq \alpha(\ell + 1)$ and ℓ sufficiently large. Hence $P(\tau)$ has no non negative root if

$$(41) \quad 3^\ell > \frac{9}{4} \alpha^4 \frac{4}{\alpha} \ell^4.$$

However (41) will always hold for ℓ sufficiently large. Hence D_2 holds for ℓ sufficiently large. \square

Note that Theorem 6 gives Dahlquist stability for $\alpha < \alpha_1$, $k \leq \alpha(\ell + 1)$ and ℓ sufficiently large. The existence of such a result has already been suspected in [13].

The proofs of Theorem 5 and 6 are based on Corollary 2. Both results are conservative since the estimates made in Lemma 2, 3 and 4 are bad. This means that $P(\tau)$ satisfies D_2 even if k is somewhat larger than $1.5 (\ell + 1)$ respectively $\alpha_1(\ell + 1)$, but how much larger? To investigate this we have computed numerically,

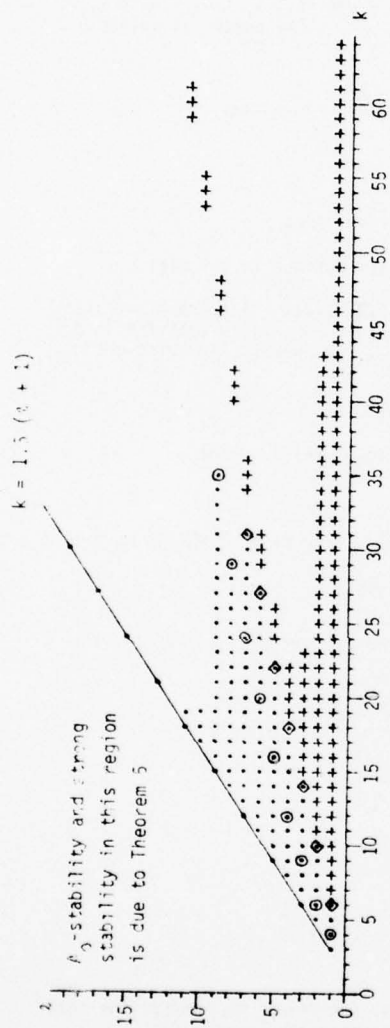


Fig. 1. Diagram of (k, l) for which Brown's (k, l) -method is A_0 -stable or not. $+$ means method is A_0 -stable; $+$ means method is not A_0 -stable since $\rho(r)$ has a root of modulus exceeding 1; \odot corresponds to (π, ρ) , see Def. 1; \otimes corresponds to (v_2, l) , see Def. 1.

using a TR 440, the numbers π_ℓ , ν_ℓ and μ_ℓ which are defined as follows.

DEFINITION 1. Let π_ℓ be such that the coefficients of $P(\tau)$ are all positive for $k \leq \pi_\ell$ and at least one is negative for $k = \pi_\ell + 1$. Let ν_ℓ be such that D_2 holds for $k \leq \nu_\ell$ and D_2 is violated for $k = \nu_\ell + 1$. Let σ_ℓ be such that Brown's (k, ℓ) -method is A_0 -stable and strongly stable for $k \leq \sigma_\ell$ but is not for $k = \sigma_\ell + 1$. Let

$$\mu_\ell = \min \{k \mid \rho(\tau) \text{ of Brown's } (k, \ell)\text{-method has a root outside the unit disk}\}.$$

Since Brown's methods always satisfy C_1 , D_1 , C_3 , D_3 one has $\pi_\ell \leq \nu_\ell \leq \sigma_\ell < \mu_\ell$. Note that one can easily show that $\pi_\ell \geq 4$ for $\ell \geq 1$. The numerical values for π_ℓ , ν_ℓ , σ_ℓ and μ_ℓ are given in Table 1.

ℓ	1	2	3	4	5	6	7	8	9	10	11
π_ℓ	4	6	9	12	16	20	24	29	35		
ν_ℓ	6	10	14	18	22	27	31				
σ_ℓ	6	10	14	18							
μ_ℓ	7	11	15	19	24	29	34	40	46	53	59

Table 1. Values of π_ℓ , ν_ℓ , σ_ℓ and μ_ℓ which are defined in Definition 1.

Note that in [13] it has been seen for $\ell = 1, 2, \dots, 11$ that Brown's (k, ℓ) -method is strongly stable for $1 \leq k \leq \mu_\ell - 1$. It is surprising that for $\ell \leq 4$ one has $\nu_\ell = \mu_\ell - 1$ and hence $\sigma_\ell = \mu_\ell - 1$. The numerical results are collected in Fig. 1.

4. PROOF OF LEMMATA. We shall need the following

LEMMA 5. Let $n \in \{1, 3\}$. Let α_n be the unique positive root of

$$(42) \quad q(\alpha) = \frac{1}{n}$$

where $q(\alpha)$ is given by (35). Then to any $\alpha < \alpha_n$ there exists $w_n(\alpha)$ such that

$$(43) \quad n \binom{k}{i-1} \frac{1}{(k-i+1)^\ell} < \binom{k}{i} \frac{1}{(k-i)^\ell}, \quad i = 1, 2, \dots, k-1$$

for $\ell \geq w_n(\alpha)$ and $k \leq \alpha(\ell + 1)$. The values for α_n as well as $w_n(\alpha)$ for some α are given in Table 2.

n	α_n	α	$w_n(\alpha)$
1	3.59112148	3.3	7
3	1.65687525	1.5	8

Table 2.

PROOF. (43) is equivalent to

$$(44) \quad f(i) = \frac{\binom{k}{i-1} (k-i+1)^{-\ell}}{\binom{k}{i} (k-i)^{-\ell}} < \frac{1}{n} \quad \text{for } i = 1, 2, \dots, k-1$$

and we ask for which values of k and ℓ is (44) satisfied. We replace the integer variable by a continuous variable s and determine the maximum of

$$f(s) = \frac{s}{k-s+1} \left(\frac{k-s}{k-s+1} \right)^{\ell}$$

for $s \in I = [1, k-1]$. $f(s)$ is a rational function with no poles in I and $f(s) > 0$ in I . From

$$f'(s) = \frac{(k-s)^{\ell-1}}{(k-s+1)^{\ell+2}} (k(k+1) - s(\ell+k+1))$$

follows that $f(s)$ has a critical point at

$$s' = \frac{k(k+1)}{\ell+k+1}.$$

This must be a maximum point since $f'(s)$ is continuous and different from zero in $[0, k] - \{s'\}$, $f'(0) > 0$ and $f'(k-\epsilon) < 0$ for $\epsilon > 0$, ϵ sufficiently small. Hence (44) is satisfied if $f(s') < 1/n$. Consider

$$g_{\ell}(k) := f(s') = \frac{k}{\ell+1} \left(\frac{k}{(k+1)(\ell+1)} \right)^{\ell}$$

which is a monotonically increasing function in k . Hence (44) is established if

$$h(\alpha, \ell) := g_{\ell}(\alpha(\ell+1)) = \alpha \left(\frac{1}{1 + \frac{1}{\ell}(1 + \frac{1}{\alpha})} \right)^{\ell}$$

satisfies

$$(45) \quad h(\alpha, \ell) < \frac{1}{n}.$$

Observe that

$$(46) \quad \lim_{\ell \rightarrow \infty} h(\alpha, \ell) = \alpha e^{-1 - \frac{1}{\alpha}} = q(\alpha).$$

Clearly $q(\alpha)$ is monotonically increasing for $\alpha \in (0, \infty)$. Hence (42) has a unique positive solution α_n . Let $\alpha < \alpha_n$ therefore $q(\alpha) < 1/n$. From (46) follows that there exists $w_n(\alpha)$ such that $h(\alpha, \ell) < 1/n$ for $\ell \geq w_n(\alpha)$ and thus (43) holds. In order to be able to compute actually $w_n(\alpha)$ given n and α , observe that for a fix α the function $h(\alpha, \ell)$ is monotonically decreasing in ℓ . Thus it is enough to verify that $h(\alpha, w_n(\alpha)) < 1/n$. \square

PROOF OF LEMMA 2. Let us write

$$S(\tau) = \sum_{i=0}^{k-1} s_i$$

where

$$(47) \quad s_i := \binom{k}{i} (k-i)^{1-\ell} \left(\frac{1+\tau}{1-\tau} \right)^i (1-\tau)^{k-1}, \quad i = 0, 1, \dots, k-1.$$

I. Let $\tau \in [0, 1]$. From (47) follows that $s_i > 0$. Hence

$$(48) \quad S(\tau) > s_{k-1} = k(1+\tau)^{k-1} \geq k(1+\tau)^{k-2} \quad \text{for } \tau \in [0, 1].$$

II. Let $\tau \in (1, \infty)$. From (43) with $n = 1$, $\alpha = 3.3 < \alpha_1$ and $w_1(\alpha) = 7$ follows that

$$(49) \quad |s_0| < |s_1| < |s_2| < \dots < |s_{k-1}| \quad \text{for } \ell \geq 8.$$

Moreover $s_i(-1)^{k-1-i} > 0$. Hence

$$(50) \quad \begin{aligned} S(\tau) &> s_{k-1} + s_{k-2} = k(1+\tau)^{k-1} + \binom{k}{2} \frac{1}{2^{\ell-1}} (1+\tau)^{k-2} (1-\tau) \\ &= (1+\tau)^{k-2} \left(\frac{k}{2^{\ell}} (2^{\ell} - k + 1) \tau + k + \frac{k(k-1)}{2} \right). \end{aligned}$$

AD-A046 552

ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK N C
PROCEEDINGS OF THE 1977 ARMY NUMERICAL AND COMPUTERS ANALYSIS C--ETC(U)
NOV 77

F/G 9/2

UNCLASSIFIED

ARO-77-3

NL

7 OF 7
ADA
046552



END
DATE
FILMED
12-77
DDC

Clearly

$$(51) \quad 2^\ell - k + 1 \geq 2^\ell - 4(\ell + 1) + 1 > 0 \quad \text{for } \ell \geq 5$$

and all k with

$$(52) \quad 1 \leq k \leq 4(\ell + 1) .$$

Hence from (51) and (50) follows

$$(53) \quad S(\tau) > k(1+\tau)^{k-2} \quad \text{for } \tau \in [1, \infty) .$$

To prove the second part of Lemma 2, we observe that (48) is valid for all k and ℓ . Moreover from Lemma 5 follows that for $\alpha < \alpha_1$ and ℓ sufficiently large (49) holds whenever $k \leq \alpha(\ell + 1)$. In (50) one has always $2^\ell - k + 1 > 0$ for ℓ sufficiently large and $k \leq \alpha(\ell + 1)$. Thus (53) holds too. \square

PROOF OF LEMMA 3. Since $|t-1| \leq |t+1|$ and $|t| < |t+1|$ for $\tau \in [0, \infty)$ one finds from (31) that

$$\begin{aligned} |T(\tau)| &\leq \sum_{j=0}^{k-3} \binom{k}{j} (k-j)^{-\ell} (1+\tau)^j \sum_{t=1}^t \binom{k-j}{2t+1} (\tau+1)^{k-1-j-2t} 4^t (\tau+1)^t \\ &= (\tau+1)^{k-2} \sum_{j=0}^{k-3} \binom{k}{j} (k-j)^{-\ell} \sum_{t=1}^t \binom{k-j}{2t+1} 4^t (\tau+1)^{1-t} . \end{aligned}$$

For $\tau \in [0, \infty)$ one has that $(\tau+1)^{1-t} \leq 1$ for all $t \geq 1$. Hence

$$\begin{aligned} \sum_{t=1}^t \binom{k-j}{2t+1} 4^t (\tau+1)^{1-t} &\leq \frac{1}{2} \sum_{t=1}^t \binom{k-j}{2t+1} 2^{2t+1} \\ &\leq \frac{1}{2} \sum_{s=0}^{k-j} \binom{k-j}{s} 2^s = \frac{1}{2} (1+2)^{k-j} \end{aligned}$$

and thus

$$(54) \quad |T(\tau)| \leq (\tau+1)^{k-2} \frac{1}{2} \sum_{j=0}^{k-3} \binom{k}{j} (k-j)^{-\ell} 3^{k-j} \quad \text{for } \tau \in [0, \infty) .$$

We apply (43) of Lemma 5 with $n = 3$, $\alpha = 1.5 < \alpha_3$ and $w_3(\alpha) = 8$. Hence

$$(55) \quad |T(\tau)| \leq (\tau+1)^{k-2} \frac{1}{2} \binom{k}{3} 3^{3-\ell} (k-2) \quad \text{for } \tau \in [0, \infty)$$

and $k \geq 3$. (55) is obviously equivalent to (34). \square

PROOF OF LEMMA 4. From (54) in the proof of Lemma 3 we have

$$(56) \quad |T(\tau)| \leq \frac{1}{2} (1+\tau)^{k-2} \sum_{i=3}^k c_{ik} \quad \text{for all } \tau \in [0, \infty) ,$$

$k \geq 1$ and $\ell \geq 1$, where

$$(57) \quad c_{ik} = \binom{k}{i} \frac{3^i}{i^\ell} , \quad i = 3, 4, \dots, k .$$

Let $\alpha \in (1, \infty)$ and we introduce

$$(58) \quad m_\ell(k) := \max \{c_{ik} \mid i = 3, 4, \dots, k\} .$$

Since $c_{ik} < c_{i, k+1}$ one has

$$(59) \quad m_\ell(k) < m_\ell(k+1) \quad \text{for all } k \text{ and } \ell .$$

Let M_ℓ be a bound for $m_\ell([a, \ell])$. Then one has

$$(60) \quad \begin{aligned} |T(\tau)| &\leq \frac{1}{2} (1+\tau)^{k-2} (k-2) m_\ell(k) \\ &\leq \frac{1}{2} (1+\tau) \alpha_\ell M_\ell \quad \text{for } \tau \in [0, \infty), k = 3, 4, \dots, [\alpha_\ell]. \end{aligned}$$

To simplify the notation we introduce

$$(61) \quad \kappa := [\alpha_\ell]$$

and

$$(62) \quad c_i := c_{i\kappa}, \quad i = 3, 4, \dots, \kappa.$$

We say that the sequence $c_i, i = 3, 4, \dots, \kappa$ has a relative maximum of order q at $i = s$ if $3 < s, s+q-1 < \kappa$

$$(63a) \quad c_{s-1} < c_s = c_{s+1} = \dots = c_{s+q-1}$$

and

$$(63b) \quad c_{s+q-1} > c_{s+q}.$$

In order to locate these relative maximas we consider

$$(64) \quad d_i = \frac{c_{i+1}}{c_i} = 3 \frac{\kappa-i}{i+1} \left(\frac{i}{i+1}\right)^\ell, \quad i = 3, 4, \dots, \kappa-1.$$

c_s is a strict relative maxima of order 1 if $d_{s-1} > 1$ and $d_s < 1$. We replace the integer variable in (64) by the continuous variable x , thus

$$(65) \quad f(x) := 3 \frac{\kappa-x}{x+1} \left(\frac{x}{x+1}\right)^\ell, \quad x \in [3, \kappa-1].$$

Since $f(0) = f(\kappa) = 0$, $f(x) > 0$ for $x \in (0, \kappa)$ and

$$(66) \quad f'(x) = 3(\kappa\ell - x(\ell+1+\kappa)) \frac{x^{\ell-1}}{(x+1)^{\ell+2}},$$

$f(x)$ has exactly one maximum point in $(0, \kappa)$, namely at

$$(67) \quad \bar{x} = \frac{\kappa\ell}{\ell+1+\kappa}.$$

Hence if $f(\bar{x}) \leq 1$ then the sequence has no relative maximum and

$$(68) \quad m_\ell([a, \ell]) = c_3.$$

If $f(\bar{x}) \geq 1$ then there are two numbers x^- and x^+ such that $0 < x^- < \bar{x} < x^+ < \kappa$ and $f(x^-) = f(x^+) = 1$, $f'(x^-) > 0$, $f'(x^+) < 0$. Assume further that

$$(69) \quad x^+ - x^- > 1 \quad \text{and} \quad \kappa - x^+ > 1$$

then the sequence c_i has exactly one relative maxima, namely c_s , of order q , where

$$(70) \quad \begin{aligned} s &= x^+, \quad q = 2 \quad \text{if } x^+ \in \mathbb{N} \\ s &= [x^+] + 1, \quad q = 1 \quad \text{if } x^+ \notin \mathbb{N}. \end{aligned}$$

Since everything has to be done for ℓ large it is more convenient to normalize the interval $(0, \kappa)$ introducing the new variable

$$(71) \quad t = \frac{x}{\alpha \ell}.$$

Hence

$$g_{\ell}(t) := f(\alpha \ell t) := 3 \frac{\frac{\kappa}{\alpha \ell} - t}{t + \frac{1}{\alpha \ell}} \left(1 - \frac{1}{\alpha \ell t + 1}\right)^{\frac{1}{\alpha}}.$$

Since $(\alpha \ell - 1)/\alpha \ell < \kappa/\alpha \ell < 1$ one has $\lim_{\ell \rightarrow \infty} \kappa/\alpha \ell = 1$. Hence

$$(72) \quad \lim_{\ell \rightarrow \infty} g_{\ell}(t) = 3 \frac{1-t}{t} e^{-\frac{1}{\alpha t}} =: g(t), \quad t \in (0, \infty).$$

Let $\bar{t}_{\ell} = \bar{x}/\alpha \ell$. Then using (67) we find

$$(73) \quad \lim_{\ell \rightarrow \infty} \bar{t}_{\ell} = \lim_{\ell \rightarrow \infty} \frac{\frac{\kappa}{\alpha \ell}}{1 + \frac{1}{\ell} + \frac{\kappa}{\ell}} = \frac{1}{1+\alpha} =: \bar{t}.$$

\bar{t} is the unique maximum point of $g(t)$, as can be verified directly from (72). Since

$$g(\bar{t}) = g(1/(1+\alpha)) = 3 \alpha e^{-1 - \frac{1}{\alpha}}$$

is a monotonic increasing function in α there is exactly one positive α with

$$(74) \quad g(1/(1+\alpha)) = 1.$$

$\alpha = \alpha_3$, as given in Table 2, is the positive root of (74), since (74) is identical to (42) with $n = 3$. Hence for any $1 < \alpha < \alpha_3$ there exists a positive L_{α} such that one has

$$(75) \quad m_{\ell}([\alpha \ell]) = c_3 \quad \text{for } \ell \geq L_{\alpha}.$$

Now let $\alpha > \alpha_3$. Let

$$(76) \quad t_{\ell}^{-} = x^{-}/\alpha \ell, \quad t_{\ell}^{+} = x^{+}/\alpha \ell.$$

Then the limits of t_{ℓ}^{-} and t_{ℓ}^{+} as ℓ tends to infinity exist. Let

$$(77) \quad t^{-} := \lim_{\ell \rightarrow \infty} t_{\ell}^{-}, \quad t^{+} := \lim_{\ell \rightarrow \infty} t_{\ell}^{+}$$

and thus

$$(78) \quad 0 < t^{-} < \bar{t} < t^{+} < 1$$

and

$$(79) \quad g(t^{-}) = g(t^{+}) = 1 \\ g'(t^{-}) > 0, \quad g'(t^{+}) < 0.$$

In order to locate t^{+} observe that t^{+} depends on α . Thus denote it by $t^{+}(\alpha)$. $t^{+}(\alpha)$ is given implicitly by (79), that is

$$(80) \quad 3 \frac{1-t^{+}(\alpha)}{t^{+}(\alpha)} e^{-\frac{1}{t^{+}(\alpha)}} = 1.$$

Implicit differentiation of (80) shows using (78) that

$$\frac{dt^{+}(\alpha)}{d\alpha} > 0 \quad \text{for } \alpha \in (\alpha_3, \infty).$$

Hence

$$(81) \quad \lim_{\alpha \rightarrow \alpha_3^+} t^+(\alpha) < t^+(\alpha) < \lim_{\alpha \rightarrow \infty} t^+(\alpha) .$$

However from (77) and (73) follows that

$$(82) \quad \lim_{\alpha \rightarrow \alpha_3^+} t^+(\alpha) \geq \frac{1}{1+\alpha_3} = 0.3763819923 .$$

Since

$$\lim_{\alpha \rightarrow \infty} g(t) = 3 \frac{1-t}{t} \quad \text{for } t \in (0, \infty)$$

we find for $t_0 := \lim_{\alpha \rightarrow \infty} t^+(\alpha)$ the equation

$$(83) \quad 3 \frac{1-t_0}{t_0} = 1 .$$

From (83) we find $t_0 = 3/4$ and this gives together with (81) and (82)

$$\frac{1}{1+\alpha_3} < t^+(\alpha) < 3/4 .$$

Using (77) we see that for any $\alpha \in (\alpha_3, \infty)$ there exists L'_α such that for all $\ell \geq L'_\alpha$ one has

$$\frac{1}{1+\alpha_3} < t_\ell^+ < 3/4$$

and hence by (76)

$$(84) \quad \frac{1}{1+\alpha_3} \alpha \ell < x^+ < \frac{3}{4} \alpha \ell .$$

From (78), (77), (83) together with (76) follows that there exists to any $\alpha \in (\alpha_3, \infty)$ a positive $L''_\alpha \geq L'_\alpha$ such that (69) is satisfied for all $\ell \geq L''_\alpha$. From (70) one sees that there exists to any $\alpha \in (\alpha_3, \infty)$ a positive $L'''_\alpha \geq L''_\alpha$ such that any relative maxima c_s of the sequence c_s satisfies

$$(85) \quad 0.3 \alpha \ell < s < 0.8 \alpha \ell$$

for $\ell \geq L'''_\alpha$. For $\alpha = \alpha_3$ it might be that c_j has a relative maximum even though $g(t) \leq 1$. However from

$$\lim_{\ell \rightarrow \infty} \bar{t}_\ell = \frac{1}{1+\alpha_3} = 0.376$$

follows that there exists L'''_{α_3} such that for $\ell > L'''_{\alpha_3}$ a possible relative maxima c_s satisfies (85) with α replaced by α_3 . Since x^+ corresponds to the only possible minimum of c_j one has that for $\alpha \in [\alpha_3, \infty)$ and $\ell \geq L'''_\alpha$

$$m_\ell(\kappa) = \max \{c_j, c_j \mid 0.3 \alpha \ell < j < 0.8 \alpha \ell\} .$$

We shall show that for ℓ large enough one has $m_\ell(\kappa) = c_3$. From (62) and (57) we find

$$c_i = \binom{\kappa}{i} \frac{3^i}{i^\ell} = \frac{\kappa!}{i!(\kappa-i)!} \frac{3^i}{i^\ell} = \frac{\Gamma(\kappa+1)}{\Gamma(i+1)\Gamma(\kappa-i+1)} \cdot \frac{3^i}{i^\ell} ,$$

where $\Gamma(x)$ is the Gamma function. Hence for j with $j/\alpha \ell \in [0.3, 0.8]$ and ℓ sufficiently large one has

$$(86) \quad \frac{c_j}{c_3} = \frac{3!}{\Gamma(j+1)\Gamma(\kappa-j+1)} \frac{3^{j-3+\ell}}{j^\ell} \leq K_1 \frac{\Gamma(\alpha \ell - 2)}{\Gamma(j+1)\Gamma(\alpha \ell - j)} \frac{3^{j+\ell}}{j^\ell} ,$$

where $K_1 = 2/9$. Here we used that the Gamma function $\Gamma(x)$ is monotonically in-

creasing for $x \geq 1$ and that $\alpha \ell - j \geq \alpha \ell 0.2 > 1$ for ℓ sufficiently large. For all $x \geq 1$ we can use the crude version of Stirlings formula

$$(87) \quad \frac{1}{2} \left(\frac{x}{e}\right)^x \sqrt{2\pi x} \leq \Gamma(x+1) \leq 2 \left(\frac{x}{e}\right)^x \sqrt{2\pi x}$$

Using (87) in (86) one finds

$$(88) \quad \frac{c_j}{c_3} \leq K_2 \sqrt{\frac{1-t-\frac{1}{\alpha \ell}}{t(1-\frac{3}{\alpha \ell})}} \frac{B_\ell^\ell (\alpha \ell)^{-\ell-2.5}}{5}$$

where $t = j/\alpha \ell$,

$$K_2 = \frac{8}{\sqrt{2\pi}} e^2 K_1$$

and

$$B_\ell^\ell = \frac{(1-\frac{3}{\alpha \ell})^{\alpha} 3^{t\alpha+1}}{t^{t\alpha+1} (1-t-\frac{1}{\alpha \ell})^{\alpha-t\alpha}}$$

For $\alpha \ell > \max\{\frac{2}{1-t}, 6\}$ we have

$$(89) \quad B_\ell^\ell < B = \frac{3^{t\alpha+1}}{t^{t\alpha+1} (\frac{1-t}{2})^{\alpha-t\alpha}}$$

and

$$(90) \quad \frac{c_j}{c_3} \leq 32 K_2 \sqrt{\frac{1-t}{t}} \left(\frac{B}{\alpha}\right)^\ell \frac{1}{\ell}$$

Since B and α are independent of ℓ it follows from (90) and (89) that for any $\alpha \in [\alpha_3, \infty)$ there exists $L_\alpha \geq L_\alpha'''$ such that

$$\frac{c_j}{c_3} < 1 \quad \text{for } t = \frac{j}{\alpha \ell} \in [0.3, 0.8], \quad \ell \geq L_\alpha$$

Hence

$$(91) \quad m_\ell(\{\alpha \ell\}) = c_3 \quad \text{for } \ell \geq L_\alpha$$

However from (62), (61) and (57) one has

$$(92) \quad c_3 = \binom{\ell \alpha \ell}{3} \frac{3^3}{3^\ell} \leq \frac{9}{2} \alpha^3 \ell^3 3^{-\ell} =: M_\ell$$

(60) together with (75), (91) and (92) give the desired estimate for all $\ell \geq L_\alpha$, $k \leq \alpha \ell$. \square

REFERENCES

1. Brown, R.L.: Multi-derivative numerical methods for the solution of stiff ordinary differential equations. Department of Computer Science, University of Illinois, Report UIUCDCS-R-74-672, 1974.
2. -----: Some characteristics of implicit multistep multi-derivative integration formulas. To appear in SIAM J. on numer. Analysis.
3. Creedon, D.M. and Miller, J.J.: The stability properties of a q-step backward differences schemes. BIT 15, 1975, 244-249.
4. Cryer, C.W.: On the instability of high order backward-difference multistep methods. BIT 12, 1972, 17-25.
5. -----: A new class of highly-stable methods: A_0 -stable methods. BIT 13, 1973, 153-159.

6. Dickson, L.E.: New first course in the theory of equations. John Wiley, New York-London (1962).
7. Gear, C.W.: The automatic integration of stiff ordinary differential equations. Information Processing 68, North Holland Publ. Co., Amsterdam, (1969), 187-193.
8. Griepentrog, E.: Mehrschrittverfahren zur numerischen Integration von gewöhnlichen Differentialgleichungssystemen und asymptotische Exaktheit. Wiss. Z., Humboldt-Univ. Berlin Math.-Natur. Reihe 19, 1970, 637-653.
9. Henrici, P.: Discrete variable methods in ordinary differential equations. New York - London, J. Wiley & Sons, 1962.
10. Jeltsch, R.: Multistep multiderivative methods and Hermite-Birkhoff interpolation. Proceedings of the Fifth Manitoba Conference on numerical mathematics. Oct. 1-4, 1975, Congressus Numerantium XVI, Utilitas Mathematica Publishing Incorporated, Winnipeg (1976), 417-428.
11. -----: Stiff stability and its relation to A_0 - and $A(0)$ -stability. SIAM J. Numer. Anal., 13, (1976), 8-17.
12. -----: Stiff stability of multistep multiderivative methods. SIAM J. Numer. Anal., to appear in vol. 14, Sept. 1977.
13. -----/Kratz, L.: On the stability properties of Brown's multistep multiderivative methods. to appear.
14. Liniger, W.: Zur Stabilität der numerischen Integrationsmethode für Differentialgleichungen. Doctoral thesis, University of Lausanne, Switzerland (1957).
15. Spijker, M.N.: Convergence and stability of step-by-step methods for the numerical solution of initial-value problems. Numer. Math., 8, 1966, 161-177.
16. Stetter, H.J.: Analysis of Discretization methods for ordinary differential equations. New York - Heidelberg - Berlin, Springer, 1973.
17. Werner, H.: Praktische Mathematik I. Springer Berlin - Heidelberg - New York (1970).

AN EXAMPLE OF APPARENT CONVERGENCE TO THE WRONG LIMIT

Walter Gautschi
Mathematics Research Center
University of Wisconsin-Madison
Madison, Wisconsin 53706

ABSTRACT. One owes to Perron the continued fraction

$$\frac{M'(a,b;z)}{M(a,b;z)} = \frac{a}{b-z} \frac{(a+1)z}{b+1-z} \frac{(a+2)z}{b+2-z} \dots$$

for the logarithmic derivative of Kummer's function. It is pointed out here that the continued fraction exhibits a strong phenomenon of apparent convergence to the wrong limit, when $|z|$ is large and $\operatorname{Re} z \geq 0$. The phenomenon, in a weakened form, may persist when $\operatorname{Re} z < 0$, but disappears for $z = -x < 0$ and $0 < a + 1 \leq b$. The matter is further illustrated in the special cases of Bessel functions ($a = \nu + 1/2$, $b = 2\nu + 1$) and incomplete gamma functions ($b = a + 1$). The complete paper under the title "Anomalous convergence of a continued fraction for ratios of Kummer functions" is available as MRC Technical Summary Report #1711, January 1977, and is to appear in Mathematics of Computation, October 1977.

**FLOATING POINT COMPUTATION FACILITIES
FOR A
COMMON PROGRAMMING LANGUAGE FOR THE DOD**

David A. Fisher
Science and Technology Division
Institute for Defense Analyses
Arlington, Virginia 22202

ABSTRACT. This paper discusses some of the considerations and tradeoffs that led to the technical requirements for the floating point facilities for a common programming language for the DoD. Some implications for the requirements in designing and implementing a language are also given.

1. INTRODUCTION. The Department of Defense (DoD) is attempting to limit the number of general purpose programming languages used in new defense systems. Currently most data processing in the DoD is programmed using COBOL and most scientific applications using FORTRAN. There is no intention to change this situation.

Most of the costs and problems of software in the DoD, however, are associated with embedded computer systems. An embedded computer system is integral to a larger system such as an electromechanical device, combat weapon system, tactical system, aircraft, ship, missile, spacecraft, command and control system, communication system, or other system whose primary function is not computation. It also includes the support software for the design, development, and maintenance of such systems. Data processing, scientific, and research computers are not normally included among embedded computer systems.

Currently, there are no widely used languages for embedded computer systems. Major benefits to the DoD are expected if the number of languages used in embedded computer applications can be reduced. Programming languages are neither the cause of nor the solution to software problems, but because of their central role in all software activity, they can either aggravate existing problems or simplify their solutions.

In determining an appropriate common programming language, the DoD has used the following three major selection criteria:

- The language should support unique characteristics of embedded computer systems. Specifically, software for embedded computer systems must be able to interface with nonstandard input-output devices, must be able to respond to unplanned error situations,

must be able to meet the real-time constraints for the applications and devices, and must be able to perform several tasks concurrently.

- The language should support the needs of the software environment for defense systems. Embedded computer systems are often large, long-lived, and subject to continuous change. They must conform to the physical and real-time constraints of the associated hardware. Software reliability and automatic recovery from failures are often critical to defense systems.
- The language will be suitable as a common language. Its object programs must run on a variety of computers, its definition must be complete and unambiguous, its users must be able to specify their programs in terms of the applications (rather than in terms of their object representation), and the language must have a stable definition with enforceable standards.

A recent study of 23 existing languages, including many currently used in the DoD, has shown that none is appropriate as a common programming language for embedded computer systems. The Services have, therefore, undertaken a joint effort to obtain a suitable language through programming language design and/or modification. As a first step they have agreed upon a set of requirements or desired characteristics for a common language. The requirements reflect the major criteria at a more language-oriented level without specifying individual language features.

No language has yet been chosen as a common DoD language and no determination has been made for the numeric facilities of such a language. The technical requirements are a consensus of the needs of the Services. Any examples given below are given as illustrations of features that satisfy the requirements. They are not necessarily indicative of features that ultimately will be selected.

This paper reviews some of the considerations used to determine the technical requirements for the floating point and fixed point facilities. The requirements reflect the needs of a command language for embedded computer applications and, therefore, may be inappropriate for general scientific and engineering environments.

This paper discusses requirements 3-1A through 3-1D as reported in the "Department of Defense Requirements for High Order Computer Programming Language -- Ironman", High Order Language Working Group, 14 January 1977. These requirements are concerned with the floating point facilities and with general characteristics appropriate for both floating point and fixed point computations. Syntactic issues are not discussed here; any forms used are for illustrative purposes only and do not indicate likely choices for a common language. The appendix contains the requirements likely to affect the choice of numeric facilities, including those for fixed point arithmetic (i.e., 3-1E through 3-1H).

2. GENERAL REQUIREMENTS.

3-1A. Numeric Values. The language shall provide types for integer, fixed point, and floating point numbers. Numeric operations and assignment that would cause the most significant digits of numeric values to be truncated (e.g., when overflow occurs) shall constitute an exception situation.

The common language is intended for a broad class of embedded computer applications that include sensor processing, real time control, simulation, diagnostics, counting, record keeping, and display. All these applications require numeric computation facilities in varying degrees of sophistication. The numeric requirements reflect in greater detail the general criteria (as expressed in requirements 1-A through 1-H) for a general-purpose programming language that will aid the development of reliable, maintainable, and efficient programs; that will incorporate neither unneeded generality nor unnecessary complexity; that will be practical in implementation; and that, where possible, will be machine-independent and formally defined.

Particularly important among these criteria are reliability and maintainability. It is very difficult to understand or to predict the action of programs if the most significant digits of numeric values can be inadvertently lost during a computation or assignment to storage. Some languages represent all numeric values modulo the word length of the object machine. Such a default is not only machine-dependent, but seldom represents the intent of the programmer. Consequently, the requirements specify that loss of the most significant (nonzero) digits of a numeric quantity must be accompanied by an execution time exception situation and subsequent program action specified in the program.

3. FLOATING POINT.

3-1C. Floating Point Precision. The precision of each floating point variable and expression shall be specifiable in programs and shall be determinable at translation time. Precision specifications shall be required for each floating point variable. Precision shall be interpreted as the minimum precision to be implemented in the object machine. Floating point results shall be implicitly rounded (or on some machines truncated) to the implemented precision. Explicit conversion operations shall not be required between floating point precisions.

3-1D. Floating Point Implementation. A floating point computation may be implemented using the actual precision, radix, and exponent range available in the object machine hardware. There shall be built-in operations to access the actual precision, radix, and exponent range with which floating point variables and expressions are implemented.

Floating point is a system for the approximate representation of a wide range of real numbers. They are most useful for numeric quantities whose values dynamically vary over a wide range of values, but a similar number of significant digits is desired for all values. Each floating point number can be viewed as a triple of integers: a mantissa, m ; a base, b ; and exponent, e . The value of a floating point number is mb^e . The range of values that can be represented depends primarily on the exponent range and on the base chosen. How closely a real number can be approximated depends primarily on the precision of the floating point representation. By precision is meant the number of digits in the mantissa. The base is usually implicitly represented in storage and unalterable by the user.

PRECISION. Precision is a primary concern in any system for floating point computation. The precision used affects the computational results and, therefore, should be known to the user. Because floating point hardware implementations typically provide more than one precision, the user should be able to specify the minimum precision desired in a program.

The general requirement that the language be machine independent may not be fully achievable for floating point computation, given the multiplicity of existing hardware implementations. Nevertheless, a language can be defined to avoid unnecessary machine dependencies and to minimize the effects of those that are unavoidable. Precision specifications, for example, can be made machine-independent by allowing the user to give a numerical specification of the number of digits of precision needed.

Although numeric specification of precision can be machine independent, efficiency of implementation dictates that the corresponding implementations use the precisions available in the object machine hardware. For reliability, the language must require that the implemented precision be at least as great as the specified precision. For efficiency, translators should, in each case, be permitted to use any available precision at least as large as that specified.

The desired precision often varies within a computation and even within an expression. We might, for example, want to compute the double-precision product of two single-precision variables. Consequently, precision should be specifiable for each variable and each expression (including subexpressions).

For efficient implementation, the required precision must be known at the time of translation. Because there is no particular precision that is needed most often (i.e., no appropriate default), the language should require user specification of the precision of each variable.

The desired precision for intermediate results (i.e., expressions) will usually be the maximum of the precisions of their arguments. Thus (in compliance with requirement IC), the default precision for expressions can be the maximum argument precision for the expression,

with the default precision explicitly overridden when other precisions are desired. More precisely, the default for the precision to which an operation is computed and to which intermediate results are saved should be the maximum of the implemented precisions of the arguments.

A language as described above would require explicit specification of the precision of variables, permit explicit specification of the precision of expressions, and have a default precision of all other cases. In such a language, the (implicitly or explicitly) specified precision will be known at each stage of a floating-point computation. Thus, although more than one precision may occur within a computation and, therefore, conversion between precisions will be required, all precision conversions can be implicitly specified. Conversion between precisions should be by zero fill or by rounding. Any other interpretation leads to anomalies and inconsistencies (e.g., equal becomes nontransitive).

The only detectable error situation associated with precision would be that the specified precision exceeds the maximum precision available in the object machine hardware. Such errors are detectable during translation. The language should permit either of two actions when this error occurs. Either the translator does not compile that portion of the program and generates a translation time error message, or it implements the specified precision using software routines and generates a translation time warning that the computation may be exceptionally expensive in execution (see requirements 13F and 13D, respectively).

ROUNDING RULES. The system proposed above is a tradeoff between efficiency and machine independence. It allows the maximum in execution efficiency with what appears to be machine independent specifications. The machine independence, however, depends on an assumption that rounding is used throughout the floating point computation. If truncation or other nonrounding rules are used in hardware, the significance of results will be very difficult to predict, and, without a detailed understanding of the object machine truncation rules, will be impossible.

The language should not permit the use of floating-point hardware with unusual rounding rules to implement the standard floating point facilities of the language. Nonstandard floating point might be used in the language as a user-defined type (see requirement 3C) that is defined as a machine-dependent feature (see requirement 11D).

TRANSLATION TIME FUNCTIONS. Even with rounding of floating point results and explicit numeric specifications of precision, programs will not be completely machine independent. The results of a floating point computation will depend on the precision, base, and exponent range used in the implementation. Programs that account for differences in precision, base, and exponent range can be described in a machine-independent form if the language provides translation time functions to access the implemented precision, base, and range.

Precision could be accessed by a function that applies to any variable or expression and returns the actual precision used to implement that variable or function. Normally, there is a single choice of radix for a given machine, so the base could be accessed by a single parameterless translation time function.

The exponent range is the major factor in determining the range of values that can be represented in a floating point representation. The range of values, rather than the exponent range itself, however, is of greater use to the user. The range of values depends on precision as well as exponent range. This suggests a need for two translation time functions, each taking a precision as their argument. A maximum-value function might return the maximum positive value representable in the implementation of the specified precision, while a minimum value function would return the minimum positive value representable in the corresponding implemented precision. No function is needed for the most negative representable value, since it will normally only differ in sign from the largest positive value.

FLOATING POINT OPERATIONS.

3-1B. Numeric Operations. . . There shall be built in operations for addition, subtraction, multiplication, division with floating point result, and negation for all numeric types. There shall be built-in equality (i.e., equal and unequal) and ordering operations (i.e., less than, greater than, less or equal, and greater or equal) between elements of each numeric type. Numeric values shall be equal if and only if they represent exactly the same abstract value. The semantics of all built-in numeric operations shall be included in the language definition. [Note that there might also be standard library definitions for numeric functions such as exponentiation.]

Three classes of floating point operations are called for: arithmetic operations, relational operations, and library routines.

The built-in arithmetic operations will be addition, subtraction, multiplication, division, and negation. These are the operations normally provided in floating point hardware. For efficiency of implementation, the arithmetic must be implemented using the available hardware operations and, therefore, cannot be completely defined in a language that must produce codes for several machines. The language definition can, however, specify properties that must be met by any correct implementation.

The language definition might require that any single floating point addition approximate real addition within specified precision. It might require that addition be commutative, and that it be nondecreasing in positive arguments. The language can guarantee reasonableness of the operations while excluding few floating-point implementations.

Equal, unequal, greater, less, greater or equal, and less or equal floating point relational operations will be provided. Because of approximate nature of floating-point arithmetic, equality between floating-point numbers will not necessarily imply equality among the corresponding real numbers. As with the arithmetic operations, the language can dictate certain properties for the relational operations. They should, except for unequal, be transitive; equal and unequal should be commutative; and equal should be reflexive. Notice that equal between arguments of different implemented precisions will be transitive only if the lesser precision is expanded to the greater precision using zero fill before the comparison takes place. This is consistent with the default precision of expressions mentioned earlier, and implies that floating numbers will be equal if, and only if, *their normalized representations in the largest available precision are identical.*

The language is to support library facilities (see requirement 12A) and it is expected that standard library definitions for mathematical functions will be made available. Users of any floating point facility should realize that, in general, a subroutine cannot produce results to the precision of the operations comprising its body. This means that library routines will generally have to do internal computations to a precision greater than that specified for their results.

**APPENDIX. TECHNICAL REQUIREMENTS AFFECTING
THE NUMERIC COMPUTATION FACILITIES**
(excerpted from the Ironman document)

The technical requirements for a common DoD high order programming language are a synthesis of the requirements submitted by the Military Departments. They specify a set of language characteristics that are appropriate for embedded computer applications (i.e., command and control, communications, avionics, shipboard, test equipment, software development and maintenance, and support applications). The subset of the requirements reproduced here are those most likely to influence the choice of numeric computation facilities.

General Syntax

2B. Grammar. The language should have a simple, uniform, and easily parsed grammar and lexical structure. The language shall have free form syntax and should use familiar notations where such use does not conflict with other goals.

2C. Syntactic Extensions. The user shall not be able to modify the source language syntax. In particular the user shall not be able to modify or introduce new precedence rules or to define new syntactic forms.

2G. Numeric Literals. There shall be built-in numeric literals. Numeric literals shall have the same values in programs as in data.

Types

3A. Strong Typing. The language shall be strongly typed. That is, the type or mode of each variable, array and record component, expression, function, and parameter shall be determinable at translation time.

3B. Implicit Type Conversions. There shall be no implicit conversions between types.

3C. Type Definitions. It shall be possible to define new data types in programs. Type definitions shall be processed entirely at translation time. The scope of a type definition shall be determinable at translation time. No restriction shall be imposed on defined types unless it is imposed on all types.

Numeric Types

3-1A. Numeric Values. The language shall provide types for integer, fixed point, and floating point numbers. Numeric operations and assignment that would cause the most significant digits of numeric values to be truncated (e.g., when overflow occurs) shall constitute an exception situation.

3-1B. Numeric Operations. There shall be built-in operations (i.e., functions) for conversion between numeric types. There shall be built-in operations for addition, subtraction, multiplication, division with floating point result, and negation for all numeric types. There shall be built-in equality (i.e., equal and unequal) and ordering operations (i.e., less than, greater than, less or equal, and greater or equal) between elements of each numeric type. Numeric values shall be equal if and only if they represent exactly the same abstract value. The semantics of all built-in numeric operations shall be included in the language definition. [Note that there might also be standard library definitions for numeric functions such as exponentiation.]

Floating Point Type

3-1C. Floating Point Precision. The precision of each floating point variable and expression shall be specifiable in programs and shall be determinable at translation time. Precision specifications shall be required for each floating point variable. Precision shall be interpreted as the minimum precision to be implemented in the object machine. Floating point results shall be implicitly rounded (or on some machines truncated) to the implemented precision. Explicit conversion operations shall not be required between floating point precisions.

3-1D. Floating Point Implementation. A floating point computation may be implemented using the actual precision, radix, and exponent range available in the object machine hardware. There shall be built-in operations to access the actual precision, radix, and exponent range with which floating point variables and expressions are implemented.

Integer and Fixed Point Types

3-1E. Integer and Fixed Point Numbers. Integer and fixed point numbers shall be treated as exact numeric values. There shall be no implicit truncation or rounding in integer and fixed point computations.

3-1F. Integer and Fixed Point Variables. The range of each integer and fixed point variable must be specified in programs and determinable at translation time. Such specifications shall be interpreted as the minimum range to be implemented. Explicit conversion operations shall not be required between numeric ranges.

3-1G. Fixed Point Scale. The scale or step size (i.e., the minimal representable difference between values) of each fixed point variable must be specified in programs and be determinable at translation time.

3-1H. Integer and Fixed Point Operations. There shall be built-in operations for integer and fixed point division with remainder and for conversion between fixed point scale factors. The language shall require explicit scale conversion operations whenever the scale of a value must be changed to properly perform some operation (e.g., assignment, comparison, or parameter passing).

Composite Types

3-3A. Composite Type Definitions. It shall be possible to define types that are Cartesian products of other types. Composite types shall include arrays (i.e., composite data with indexable components of homogeneous types) and records (i.e., composite data with labeled components of heterogeneous type). **3-3D. Array Specifications.** The number of dimensions for each array must be specified in programs and shall be determinable at translation time. The range of subscript values for each dimension must be specified in programs and shall be determinable by the time of array allocation. The range of subscript values shall be restricted to a contiguous sequence of integers or to a contiguous sequence from an enumeration type. [Note that translators may be able to produce more efficient object programs where subscript ranges are determinable at translation time.]

3-3E. Operations on Subarrays. There shall be built-in operations for value access, assignment, and catenation of contiguous sections of one-dimensional arrays of the same component type.

Expressions

4A. Form of Expressions. The form (i.e., context free syntax) of expressions shall not depend on the types of their operands or on whether the types of the operands are built into the language.

4B. Type of Expressions. The language shall require that the type of each expression be determinable at translation time. It shall be possible to specify the type of an expression explicitly. [Note that the latter requirement provides a way to resolve ambiguities in the types of literals and to assert the type of results; it does not provide a mechanism for type conversion.]

4C. Side Effects. The language should permit few side effects in expressions. In particular, during expression evaluation assignment shall not be allowed to any variable that is accessible in the scope of the expression.

4D. Allowed Usage. Expressions of a given type shall be allowed wherever both constants and variables of the type are allowed.

4E. Constant Valued Expressions. Constant valued expressions (i.e., expressions whose values are determinable at translation time) shall be allowed wherever constants of the type are allowed. Such expressions shall be evaluated before execution time.

4F. Operator Precedence Levels. The precedence levels (i.e., binding strengths) of all infix operators shall be specified in the language definition, shall not be alterable by the user, shall be few in number, (e.g., three or four), and shall not depend on the types of the operands. [Note that there might be built-in operator symbols whose meaning is entirely specified by the user.]

4G. Effect of Parentheses. Explicit parentheses shall dictate the association of operands with operators. Explicit parentheses shall be required to resolve the operator-operand associations wherever an expression has a nonassociative operator to the left of an operator of the same precedence.

Parameters

7H. Formal Array Parameters. The number of dimensions for formal array parameters must be specified in programs and shall be determinable at translation time. Determination of the subscript range for formal array parameters may be delayed until execution and may vary from call to call. Subscript ranges shall be accessible within function and procedure bodies without being passed as an explicit argument.

Specifications of Object Representation

11C. Machine Configuration Constants. The language shall require the declaration of certain global constants of the object machine configuration. These shall include constants that specify the machine model, the memory size, special hardware options, the operating system if present, and peripheral equipment. Such constants shall be used to

determine the object code to be generated by the translator and may also be used by the program like other constants. [Note that the user can define constants and use them as switches to control user defined compilation options.]

11D. Configuration Dependent Specifications. It shall be possible to use machine dependent facilities in programs. Portions of programs that depend on the characteristics of the object machine (e.g., on the machine model, special hardware options, device configuration, or operating system) shall be permitted only within branches of conditional control structures that discriminate on machine configuration.

1977 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE

30-31 March 1977

Sponsored by

Army Mathematics Steering Committee

U.S. Army Research Office

P. O. Box 12211

Research Triangle Park, NC 27709

Roster of Registrants

Norman Banks, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland 21005
Bruce D. Barnett, US Army - ARRADCOM/Dover, ATTN: DRDAR-MSM, Dover, New Jersey 07801
James G. Bevelock, ARRADCOM, Dover, New Jersey 07801
Paul T. Boggs, Army Research Office, P. O. Box 12211, Research Triangle Park,
North Carolina 27709
Lawrence J. Bolmarcich, Frankford Arsenal, Bridge and Tacony Streets, Philadelphia,
Pennsylvania 19137
Carl de Boer, Mathematics Research Center, University of Wisconsin-Madison,
610 Walnut Street, Madison, Wisconsin 53706
Alfred G. Brandstein, Commander, Harry Diamond Laboratories, ATTN: DRXDO-EM-2,
2800 Powder Mill Road, Adelphi, Maryland 20783
Achi Brandt, Weizmann Institute, Rehovot, Israel
Tuncer Cebeci, California State University, Long Beach, California 90801
Jagdish Chandra, US Army Research Office, P. O. Box 12211, Research Triangle Park,
North Carolina 27709
Y. F. Chang, Computer Science Department, University of Nebraska, 110 Ferguson,
Lincoln, Nebraska 68588
Peter C. T. Chen, Watervliet Arsenal, SARWV-RR-AM, Watervliet, New York 12189
John Cheng, Friends University, 2100 University Avenue, Wichita, Kansas 67213
Philippe Clement, Mathematics Research Center, University of Wisconsin-Madison,
610 Walnut Street, Madison, Wisconsin 53706
Joel S. Cohen, Mathematics Department, University of Denver, Denver, Colorado 80208
Charles C. Conley, Mathematics Department, University of Wisconsin-Madison, Madison,
Wisconsin 53706
Fred Crary, Mathematics Research Center, University of Wisconsin-Madison,
610 Walnut Street, Madison, Wisconsin 53706
Colin Cryer, Mathematics Research Center, University of Wisconsin-Madison,
610 Walnut Street, Madison, Wisconsin 53706
Elizabeth Cuthill, Naval Ship R&D Center, Bethesda, Maryland 20084
Allan Douglas, Operations Research & Analysis Establishment, National Defence
Headquarters, Ottawa, Ontario, Canada K1A 0K2
Charles S. Duris, Department of Mathematics, Drexel University, Philadelphia,
Pennsylvania 19104

Louis W. Ehrlich, Johns Hopkins Applied Physics Lab., Johns Hopkins Road, Laurel, Maryland 20810

Bo G. Einarsson, The Swedish National Defense Research Institute, Box 98, S-147 00 Tumba, Sweden

Sylvan Eisman, US Army Frankford Arsenal, Philadelphia, Pennsylvania 19137

Imre B. Emmerth, Mathematics Department, University of Wisconsin-Madison, Madison, Wisconsin 53706

Hans L. Fetter, Computer Sciences Department, University of Wisconsin-Madison, 1210 West Dayton Street, Madison, Wisconsin 53706

David A. Fisher, Institute for Defense Analysis, Arlington, Virginia

A. A. Frank, Department of Electrical Engineering, University of Wisconsin-Madison, Madison, Wisconsin 53706

Frederick N. Fritsch, Lawrence Livermore Laboratory, P. O. Box 808 (L-310), Livermore, California 94550

Arthur Garder, Southern Illinois University, Edwardsville, Illinois 62025

John C. Garth, Deputy for Electronic Technology (RADC), Hanscom Air Force Base, Massachusetts 01731

John Gary, Computer Science Department, University of Colorado, Boulder, Colorado 80302

Walter Gautschi, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

Harold R. Gehle, US Army Management Engineering Training Activity, DRXOM-SE, Rock Island, Illinois 61201

Alan C. Genz, Department of Mathematical Sciences, Northern Illinois University, De Kalb, Illinois 60115

John Gerstina, IUPUI, 1201 East 38th Street, Indianapolis, Indiana 46260

Moshe Goldberg, Department of Mathematics, University of California-Los Angeles, Los Angeles, California 90024

Marvin J. Goldstein, Naval Underwater Systems Center, New London Laboratory, New London, Connecticut 06320

Julia H. Gray, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

T. N. E. Greville, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

Louis A. Hageman, Bettis Atomic Power Laboratory, P. O. Box 79, West Mifflin, Pennsylvania 15122

Arthur Hausner, Harry Diamond Laboratories, 2800 Powder Mill Road, Adelphi, Maryland 20783

J. M. Heimerl, Ballistic Research Laboratory, DRDAR-BLB, Aberdeen Proving Ground, Maryland 21005

Morton A. Hirschberg, US Army Ballistic Research Laboratory, ATTN: DRDAR-BLB, Aberdeen Proving Ground, Maryland 21005

Jonas T. Holdeman, Jr., Union Carbide Corp., Nuclear Div., Oak Ridge National Laboratory, P. O. Box X, Oak Ridge, Tennessee 37830

David Hough, Tektronix, Inc., P. O. Box 384, Wilsonville, Oregon 97070

E. Houstis, Purdue University, West Lafayette, Indiana 47906

Ralph Hunt, Computer Operations Branch, Eustis Directorate, USAAMRDL, Ft. Eustis, Virginia 23604

M. A. Hussain, Watervliet Arsenal, Watervliet, New York 12189

L. Jedynek, Department of Electrical Engineering, University of Wisconsin-Madison, Madison, Wisconsin 53706

Rolf Jeltsch, Institute of Mathematics, Ruhr-University Bochum, D-463 Bochum, Germany

Dennis C. Jespersen, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

Sandie Jones, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

W. Kahan, University of California, Berkeley, California 94720

Eve S. Kaplan, Walter Reed Army Institute of Research, 16th Street & Alaska Ave., N.W., Washington, D. C. 20012

Patrick Keast, University of Toronto, Toronto, Ontario, Canada

Gershon Kedem, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

R. W. Klopfenstein, RCA Laboratories, P. O. Box 537, Princeton, New Jersey 08540

John H. Knight, Department of Applied Mathematics, CSIRO Australia, and University of Waterloo, Waterloo, Ontario, Canada N2L361

Michael L. Koszykowski, Department of Chemistry, University of Illinois, 173 Noyes Laboratory, Urbana, Illinois 61801

Richard Kraft, NBS, 19005 Canadian Ct., Gaithersburg, Maryland 20760

Lawrence J. Kratz, Idaho State University, Pocatello, Idaho 83209

Mark D. Kregel, Ballistic Research Laboratory, DRDAR-BLB, Aberdeen Proving Ground, Maryland 21005

Peter Lax, Courant Institute of Mathematical Sciences, New York, New York

Gary K. Leaf, Argonne National Laboratory, 9700 South Cass Ave., Argonne, Illinois 60439

Carolyn Moore Lee, The Computing Centre, University of British Columbia, 2075 Wesbrook Place, Vancouver, Canada V6T 1W5

Franklin Luk, Computer Science Department, Stanford University, Stanford, California 94305

H. I. MacDonald, US Army Airmobility R&D Laboratory, Ft. Eustis, Virginia 23604

Ralph M. McGehee, New Mexico Tech., Socorro, New Mexico 87801

James E. McTamany, US Army Coastal Engineering Research Center, Kingman Building, Ft. Belvoir, Virginia 22060

Gunter H. Meyer, School of Mathematics, Georgia Tech., Atlanta, Georgia 30332

Michael Minkoff, Argonne National Laboratory, 9700 South Cass Ave., Argonne, Illinois 60439

Harry L. Morrison, US Steel Research Laboratory, Monroeville, Pennsylvania 15146

M. Z. Nashed, University of Michigan, Ann Arbor, Michigan 48104

Kenneth Neves, Boeing Computer Service, Inc., 12824 NE 135, Kirkland, Washington 98033

Ben Noble, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

Don W. Noid, University of Illinois, 180 Noyes Laboratory, Urbana, Illinois 61801

Janes L. Noyes, FTD/ADAS Wright-Patterson AFB, Dayton, Ohio 45433

James M. Ortega, ICASE, Nasa Langley Research Center, Hampton, Virginia 23665

Jeffrey F. Painter, Department of Mathematics, University of Wisconsin-Madison, Madison, Wisconsin 53706

A. Pinkus, Mathematics Research Center, University of Wisconsin-Madison, 610 Walnut Street, Madison, Wisconsin 53706

Robert Plemmons, University of Tennessee, Knoxville, Tennessee 37916

M. Powell, University of Cambridge, DAMTP, Silver Street, Cambridge CB3 9EW, England

William D. Powers, Pacific Missile Test Center, Point Mugu, California 93042

L. B. Rall, Mathematics Research Center, University of Wisconsin-Madison,
 610 Walnut Street, Madison, Wisconsin 53706
 John R. Rice, Mathematical Sciences 428, Purdue University, West Lafayette,
 Indiana 47907
 Garry Rodrigue, Lawrence Livermore Laboratory, Livermore, California 94550
 Edwin H. Rogers, University of Waterloo, Waterloo, Ontario, Canada
 Edward Ross, US Army Natick R&D Command, Kansas Street, Natick, Massachusetts 01760
 J. Barkley Rosser, Mathematics Research Center, University of Wisconsin-Madison,
 610 Walnut Street, Madison, Wisconsin 53706
 Paul Saylor, University of Illinois-Urbana, Urbana, Illinois 61801
 George J. Schlenker, US Army Armament Materiel Readiness Command, DRSAR-SAM,
 Rock Island, Illinois 61201
 James A. Schmitt, Ballistic Research Laboratory, Aberdeen Proving Ground,
 Maryland 21005
 I. J. Schoenberg, Mathematics Research Center, University of Wisconsin-Madison,
 610 Walnut Street, Madison, Wisconsin 53706
 Maria Schonbek, Mathematics Research Center, University of Wisconsin-Madison,
 610 Walnut Street, Madison, Wisconsin 53706
 Robert Schreiber, Department of Computer Science, Yale University, 10 Hillhouse
 Ave., New Haven, Connecticut 06520
 Andrew H. Sherman, Department of Computer Science, University of Texas, Austin,
 Texas 78664
 Arthur Shieh, Mathematics Research Center, University of Wisconsin-Madison,
 610 Walnut Street, Madison, Wisconsin 53706
 A. Simkus, US Army Research Office, P. O. Box 12211, Research Triangle Park,
 North Carolina 27709
 Bruce Simpson, Department of Computer Science, University of Waterloo, Waterloo,
 Ontario, Canada
 Robert E. Singleton, US Army Research Office, P. O. Box 12211, Research Triangle
 Park, North Carolina 27709
 Robert D. Skeel, Department of Computer Science, University of Illinois, Urbana,
 Illinois 61801
 Michael L. Smith, Mathematics Research Center, University of Wisconsin-Madison,
 610 Walnut Street, Madison, Wisconsin 53706
 Philip W. Smith, Department of Mathematics, Texas A&M University, College
 Station, Texas 77843
 Shirley Janet Smith, Army Aviation Systems Command, P. O. Box 209, DRSAB-DA,
 St. Louis, Missouri 63166
 Royce Soanes, Watervliet Arsenal, SARWV-RR-C, Watervliet, New York 12189
 Frank Stenger, University of British Columbia, Vancouver, British Columbia, Canada
 Kunio Tanabe, North Carolina State University and Institute for Statistical
 Mathematics, 1231 Kingston Ridge Road, Cary, North Carolina 27511
 S. M. Taylor, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland 21005
 Dennis Tracey, US Army Materials and Mechanics Research Center, Watertown,
 Massachusetts 02172
 Ronald P. Uhlig, HQ, US Army Materiel Development and Readiness Command,
 5001 Eisenhower Avenue, Alexandria, Virginia 22333
 Richard Underwood, General Electric Co., San Jose, California
 H. A. van der Vorst, ACCU, Budapestlaan 6, de uithof - Utrecht, Netherlands
 John Van Rosendale, Department of Computer Science, University of Illinois,
 31 F Digital Computer Laboratory, Urbana, Illinois 61801

Cristobal Vargas, Mathematics Research Center, University of Wisconsin-Madison,
610 Walnut Street, Madison, Wisconsin 53706

Chia Ping Wang, US Army Natick R&D Command, Natick, Massachusetts 01760

Gerald L. Webb, European Space Agency, ESTEC, Doremweg, Noordwijk, Holland

Stanley Woolf, ARCON Corporation, 18 Lakeside Office Park, Wakefield,
Massachusetts 01880

J. M. Yohe, Mathematics Research Center, University of Wisconsin-Madison,
610 Walnut Street, Madison, Wisconsin 53706

Csaba K. Zoltani, Ballistic Research Laboratory, Aberdeen Proving Ground,
Maryland 21005

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARO Report 77-3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proceedings of the 1977 Army Numerical Analysis and Computers Conference		5. TYPE OF REPORT & PERIOD COVERED Interim Technical Report
7. AUTHOR(s)		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Army Mathematics Steering Committee on behalf of the Chief of Research, Development and Acquisition		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Army Research Office ATTN: DRXRO-MA P.O. Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE November 1977
		13. NUMBER OF PAGES 601
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release, distribution unlimited The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES This is a technical report resulting from the 1977 Army Numerical Analysis and Computers Conference. It contains papers on computer aided design and engineering as well as papers on numerical analysis.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <div style="display: flex; justify-content: space-between;"> <div> finite element method free boundary problems capacitance matrix method harmonic mixed boundary value problems parabolic partial differential equations the "Sinc-Galerkin" method hyperbolic problems numerical methods: ELL PACK storage and retrieval methods a parallel array computer software for interval arithmetic automatic differentiation boundary layer problems VBV-splines </div> <div> numerical quadrature Rayleigh-Ritz approximation gun tube problems in elastic-plastic range perturbation methods vector computers application of Texas instrument SR-52 decomposition of sparse matrices coupled stiff differential equations multistep multiderivative methods convergence to the wrong limit a common programming language for DOD </div> </div>		

★U.S. GOVERNMENT PRINTING OFFICE: 1977-743-666

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

603

Unclassified

PRECEDING PAGE BLANK-NOT FILMED